Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Uroš Bajc

Razvoj spletne rešitve za modeliranje postopkov v telemedicini

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: prof. dr. Marko Bajec

Ljubljana, 2017

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

V diplomskem delu preučite področje modeliranja tele-medicinskih obravnav, to je postopkov oddaljenega zajema pacientovih zdravstvenih podatkov. V okviru oddaljenega spremljanja mora namreč zdravnik pacientu predpisati, kako pogosto in ob kakšnih pogojih naj si posamezne zdravstvene podatke meri. V ta namen mora imeti zdravnik na voljo informacijsko podporo, ki je intuitivna in enostavna za uporabo in omogoča zajem takšnega postopka. Glede na vaše ugotovitve, do katerih boste prišli v okviru preučevanja obstoječih rešitev in področja oddaljenega spremljanja pacientov nasploh, razvijte spletno aplikacijo, ki bo podprla po vaši oceni najbolj smiseln način modeliranja tele-medicinskih obravnav. Pri tem upoštevajte standarde, ki obstajajo na področju zdravstva.

Za strokovno pomoč pri pisanju diplomske naloge se najprej zahvaljujem prof. dr. Marku Bajcu. Hvala tudi asistentoma Marku Jankoviću in dr. Slavku Žitniku za koristne nasvete, ki sta mi jih nudila pri razvoju spletne aplikacije.

Zahvaliti se moram še staršem, bratu in sestri, ki so mi tekom študija stali ob strani ter me podpirali. V zahvali pa ne smem pozabiti omeniti vseh prijateljev, še posebej sostanovalcev in sosedov v študentskem domu, ki so mi mnogokrat popestrili in polepšali študentsko življenje.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
REST	REpresentational State Transfer	predstavitveni prenos stanja
MVC	Model-View-Controller	model-pogled-kontroler
API	Application Programming Interface	aplikacijski vmesnik (skupek protokolov in orodij za ustvarjanje spletnih aplikacij)
HTTP	Hyper-Text Transfer Protocol	metoda za prenos podatkov po medmrežju
IoT	Internet of Things	internet stvari
XML	Extensible Markup Language	meta-označevalni jezik, namenjen opisovanju strukturiranih podatkov
JSON	JavaScript Object Notation	JavaScript zapis objektov
URI	Uniform Resource Identifier	univerzalni enolični identifikator
HTML	hypertext markup language	jezik za označevanje nadbesedila
CSS	Cascading Style Sheets	kaskadne slogovne pole
FHIR	Fast Health Interoperability Resources	hitri zdravstveni viri, namenjeni povezovanju
LCD	Liquid Crystal Display	zaslon s tekočimi kristali

SOAP	Simple Object Access Protocol	enostaven protokol za dostop do objektov
ACID	Atomicity, Consistency, Isolation, Durability	atomarnost, konsistentnost, izolacija/neodvisnost, trajnost
ORM	Object-relational mapping	objektno-relacijsko mapiranje
CRUD	Create, Read, Update and Delete	ustvari, preberi/pridobi, posodobi, izbriši
WYSIWYG	What You See Is What You Get	Tak kot je dokument na zaslonu, takšen bo tudi na papirju, internetu.
CHF	Congestive Heart Failure	kronično srčno popuščanje
BMI	Body Mass Index	indeks telesne mase

Povzetek

Naslov: Razvoj spletne rešitve za modeliranje postopkov v telemedicini

Avtor: Uroš Bajc

Oddaljeno spremljanje bolnikov na domu omogoča pacientom zmanjšanje števila obiskov pri zdravniku in dvig kvalitete njihovega življenja. Ker je fizične obiske težko nadomestiti z virtualnimi pregledi, se diplomsko delo osredotoča na modeliranje zdravniških pregledov, ki jih pacienti opravljajo na domu. Cilj diplomske naloge je tako bil izdelati prototip spletne aplikacije, s katero je mogoče ustvarjati zdravniške preglede. Največ pozornosti je bilo namenjene načinu modeliranja pregledov in njihovi predstavitvi v formatu, ki omogoča čim bolj učinkovito izmenjavo in razčlenjevanje. V diplomi so zato najprej predstavljene že obstoječe rešitve, ki naslavljajo enako ali podobno tematiko. Sledi njihova primerjava, ki služi kot osnova za načrt aplikacije, prikazan v nadaljevanju. Nato je opisana arhitektura aplikacije, na koncu pa so navedene še nadgradnje in morebitne izboljšave ter evalvacija rešitve. V diplomskem delu so tako predstavljeni principi, ki jih je priporočljivo upoštevati pri razvoju aplikacij s takim namenom. Razvita aplikacija pa služi kot zgled sledenja tem principom v praksi.

Ključne besede: zdravstvo, telemedicina, oddaljena oskrba bolnikov, modeliranje postopkov.

Abstract

Title: Developing a web application for modelling of procedures in telemedicine

Author: Uroš Bajc

Benefits of remote patient monitoring include decreased number of patients' hospital visits and an improvement in their quality of living. Physical examinations are hardly fully replaced by virtual examinations. Therefore, our thesis is focused on modelling of virtual medical examinations, which are performed by patients at their homes. The main goal of this work was to develop a prototype of a web application for modelling of medical examinations. The focus was put on the way in which the examinations are modelled, as well as on their representation in the format that enables efficient exchange and parsing. Existing solutions dealing with similar subjects are described in the beginning and are followed by their comparison which serves as a basis for application plan, described in following chapters. Application architecture is presented after that. In the end, some possible application upgrades and evaluation of created solution are mentioned. Therefore, this thesis presents principles which should be taken into consideration when developing similar applications. The developed solution serves as an example.

Keywords: healthcare, telemedicine, remote patient monitoring, procedure modelling.

Poglavje 1

Uvod

V razvitih državah pričakovana življenjska doba ljudi iz desetletja v desetletje narašča. Leta 2015 je bila pričakovana življenjska doba v zahodni in severni Evropi že več kot 80 let, v celotni evropski regiji pa 76.8 let [10]. Posledica tega je, da se vztrajno veča delež starejših ljudi, ki so bolj dovzetni za obolevanje za raznimi kroničnimi boleznimi, kot so npr. srčno popuščanje, hipertenzija, sladkorna bolezen, astma ... Predvideva se namreč, da bo do leta 2050 v Evropi že kar 30% ljudi starih 65 let ali več [40]. Po drugi strani je današnji hiter in stresen način življenja vzrok za to, da v razvitih državah vse pogosteje prihaja do obolevanja ljudi za kroničnimi boleznimi že v mladosti ali v srednjih letih in tako te bolezni niso več značilne le za starejši del prebivalstva. Projekcije kažejo, da bodo v razvitih državah v prihodnjih letih kronične bolezni še vedno krivec za največji delež umrlih, predvsem pa bodo te zaradi večanja števila obolelih predstavljale vedno težje breme zdravstvenim sistemom [38]. Ker se ti v velikem številu držav že sedaj ukvarjajo s problemi, kot so daljšanje čakalnih vrst, podražitev zdravstvenih storitev, idr., bodo v prihodnje potrebne še dodatne prilagoditve in spremembe. Ena izmed rešitev, ki lahko veliko pripomore k reševanju omenjenih izzivov, je uporaba telemedicine. Izraz telemedicina ima precej širok pomen, v osnovi pa je definiran kot uporaba tehnologije za izmenjavo podatkov in dostavljanje storitev zdravstvene oskrbe na daljavo [42].

V razvoj telemedicine se zato vlaga velike vsote denarja, vrednost trga IoT v zdravstvu naj bi namreč do leta 2020 presegla 163 milijard dolarjev [24]. V zadnjih letih je poudarek predvsem na razvoju asinhronega tipa telemedicine, ki omogoča shranjevanje informacij o pacientovih vitalnih znakih in njihovo posredovanje zdravniškemu osebju. S hitrim tehnološkim napredkom in razvojem IoT postaja namreč izvajanje oddaljenjega spremljanja pacientov vse lažje, uporabniku prijaznejše in tudi cenovno ugodno.

Kljub manj razširjenem izvajanju oddaljenega spremljanja pacientov v preteklosti, se je to že do sedaj izkazalo za zelo učinkovito z vidika izobraževanja bolnikov o boleznih, ki jih pestijo, in prilagajanja oskrbe njihovim potrebam. Predvsem pa ima veliko prednost natančnejše in pogostejše spremljanje pacientovih vitalnih znakov, kar omogoča hitrejše ukrepanje ob kakršnihkoli spremembah [44]. Nekatere študije spremljanja pacientov s kroničnim srčnim popuščanjem so tudi pokazale, da oddaljeno spremljanje pacienta pozitivno vpliva tako na manjšanje pogostosti obiskov bolnikov pri zdravniku, kot tudi na zmanjšanje umrljivosti [42]. Zaradi velike količine zbranih podatkov imajo zdravniki boljšo predstavo o dejanskem stanju pacienta, kar jim omogoča lažje postavljanje diagnoz. V primeru, da imajo bolniki dostop do svojih meritev in odgovorov, lahko tako tudi sami pridobijo bolj celovit vpogled svojega zdravstvenega stanja ter lažje kontrolirajo svojo bolezen.

Uporaba telemedicine in dodeljevanja oddaljenih pregledov pacientom omogoča zdravstvenim delavcem tudi lažjo vpeljavo kliničnih poti. Te so orodje, ki zdravstvenim ekipam omogoča racionalno in na znanstvenih dokazih utemeljeno obravnavo pacienta, spremljanje opravljenega dela ter kazalnikov kakovosti, natančnejše dokumentiranje in lažjo notranjo presojo zdravstvene prakse. Njihova uporaba je že v preteklosti mnogokrat pokazala izboljšanje kakovosti in učinkovitosti dela zdravnikov ter drugih zdravstvenih delavcev [41]. Čeprav je uporaba kliničnih poti sedaj že zelo razširjena, se s prenosom kliničnih poti „s papirja“ v virtualne preglede tako zdravnike kot tudi paciente prisili k njihovemu sledenju v celoti.

Dejstvo pa je, da kljub velikemu številu dobrih lastnosti telemedicine, obstaja tudi nekaj pomanjkljivosti. Marsikateri zdravniki namreč še niso vešči uporabe informacijskih tehnologij, zato morajo biti aplikacije za oddaljeneno spremljanje pacientov take, da se jih je mogoče hitro in enostavno naučiti uporabljati. Drugi izziv je v tem, da v kolikor primerjamo fizični obisk zdravnika z virtualnim, je velika dodana vrednost prvega ta, da se tam zdravnik lahko pogovarja s pacientom ter mu zastavlja različna vprašanja, ki se lahko nanašajo na njegovo počutje, vrednosti opravljenih meritev ali na njegove prejšnje odgovore. S tem lahko hitro pridobi veliko tistih informacij, ki so nujne za postavljanje diagnoze in predpisovanje ustreznih načinov zdravljenja.

1.1 Osnovne komponente informacijskih sistemov za oddaljeno spremljanje pacientov

Informacijski sistemi, namenjeni oddaljenemu spremljanju pacientov, sestojijo iz več osnovnih komponent. Da se lahko od bolnikov pridobi meritve njihovih vitalnih znakov, so za to potrebne posebne merilne naprave in senzorji, ki to omogočajo. Ti so povezani z osrednjo napravo (npr. s tabličnim računalnikom), kjer teče aplikacija za paciente in kamor se posredujejo zajete meritve. Aplikacija skrbi za prikazovanje zdravniških pregledov, saj lahko na njej pacienti odgovarjajo na vprašanja in s pomočjo omenjenih senzorjev opravljajo različne meritve. Odgovori in vrednosti meritev se nato pošljejo na strežnik, kjer se shranijo v repozitorij kliničnih podatkov. Ponavadi lahko bolniki preko aplikacije tudi pregledujejo svoje rezultate in komunicirajo s svojimi zdravniki, bodisi z imenjavo sporočil bodisi preko video pogovora. Medicinsko osebje ima na voljo spletni portal, kjer se ustvarja nove zdravniške preglede. Vsakemu pacientu pa se nato določi, katere preglede mora opravljati na domu. Hkrati portal nudi tudi različne možnosti prikaza vseh po-

datkov, pridobljenih iz izpolnjenih pregledov, in tako zdravnikom omogoča natančen nadzor nad zdravstvenim stanjem pacientov.

1.2 Modeliranje zdravniških pregledov

V diplomskem delu smo se osredotočili na zadnjo izmed omenjenih komponent, in sicer na portal za medicinsko osebje, pri čemer je bil glavni poudarek na delu za modeliranje zdravniških pregledov. Da bi lahko z uporabo telemedicine v čim večji meri nadomestili fizične obiske bolnikov pri zdravniku, je namreč pomembno, da ima zdravstveni delavec, ki ustvarja pregled, možnost modeliranja ustreznega delovnega toka na način, ki omogoča zajem ključnih podatkov za postavljanje diagnoze oziroma za ustrezno ukrepanje v primeru odstopanj.

Oblikovanje in programiranje aplikacije za modeliranje zdravniških pregledov predstavlja precejšen izziv. V bistvu gre pri ustvarjanju zdravniških pregledov za splošen princip modeliranja procesov, s čimer se ukvarja že veliko število obstoječih rešitev, ki jih je mogoče uporabiti pri razvoju aplikacij. Problem je v tem, da so orodja za modeliranje delovnih tokov zelo splošna in ponujajo širok nabor različnih funkcionalnosti, ki za ustvarjanje zdravniških pregledov niso potrebne in bi naredile modeliranje le bolj zapleteno. Ker v diplomski nalogi stremimo k razvoju portala za medicinsko osebje, ki omogoča ustvarjanje zdravniških pregledov na način, ki je čim bolj prilagojen zdravnikom, splošne rešitve za modeliranje procesov tako ne pridejo v poštev. Po eni strani mora sicer veljati, da je tisti, ki ustvarja nov pregled, pri tem početju čim manj omejen. Omogočeno mu mora biti, da lahko naredi skoraj poljuben delovni tok. Po drugi strani pa je treba čim bolj omejiti napake, ki lahko nastanejo pri modeliranju pregledov. Za ustvarjanje mora tudi veljati, da je kar se da intuitivno in enostavno [37]. Aplikacija za kreiranje pregledov mora tako nuditi uporabniku dovolj možnosti za ustvarjanje poljubnih delovnih tokov, obenem pa uporaba njenih funkcionalnosti ne sme biti preveč zapletena in dvoumna.

1.3 Vsebina diplomske naloge

V diplomskem delu smo najprej preučili več različnih obstoječih rešitev, to je informacijskih platform, ki se ukvarjajo z oddaljenim spremljanjem pacienta na domu, in definirali njihove prednosti ter slabosti. Na podlagi ugotovitev smo nato načrtovali in razvili zdravniški portal za ustvarjanje zdravniških pregledov, namenjenih oddaljenemu spremljanju pacientov. Veliko pozornosti smo namenili tudi njihovem shranjevanju in izmenjavi med različnimi napravami, pri čemer je bil cilj postopke predstaviti v čim bolj standardnemu formatu, ki omogoča enostavno interpretacijo. Portalu smo dodali možnost nalaganja zdravniških pregledov v globalni repozitorij in prenašanje le-teh iz njega. Da bi demonstrirali, kako delujeta izmenjava in izpolnjevanje pregledov, smo implementirali tudi prototip aplikacije, namenjene pacientom, ki omogoča opravljanje pregledov in shranjevanje zajetih podatkov.

Poglavje 2

Pregled in primerjava obstoječih rešitev

Na spletu smo poizkusili najti čim več sistemov, ki se ukvarjajo z oddaljenim spremljanjem pacientov. Po pregledu obstoječih rešitev smo vsako izmed njih na kratko predstavili, pri čemer smo se osredotočili predvsem na del, namenjen ustvarjanju zdravniških pregledov. Pregledali in analizirali smo naslednje platforme:

- Medable
- Opentelehealth
- ZephyrLIFE^{TM} Home Remote Patient Monitoring System
- Alayacare
- Vivify health
- Tactio health
- Honeywell LifeCare
- Phillips eCare

Na trgu je sicer mogoče najti še druge ponudnike, ki se ukvarjajo s telemedicino. To so npr. Intel s produktom Vitalbeat, AccuHealth, idr., vendar je o njihovih produktih mogoče pridobiti zelo malo informacij. Obstajajo tudi aplikacije, ki so izrecno namenjene le vzpostavljanju video klicev z zdravniki, ne pa tudi beleženju vitalnih znakov (npr. eVisit).

2.1 Opis rešitev

2.1.1 Medable in Opentelehealth

Obe aplikaciji smo lahko natančneje preučili kot preostale, saj smo za testiranje in preučevanje Opentelehealtha imeli omogočen dostop do uporabe njihove demo aplikacije. Tako smo lahko zelo podrobno pregledali velik del funkcionalnosti, ki jih platforma nudi. Pri Medablu pa imajo na voljo podrobno dokumentacijo [16], iz katere smo lahko dobili precej dobro predstavo o delovanju aplikacije.

Opentelehealth

Namen aplikacije Opentelehealth je, da omogoča ustvarjanje zdravstvenih pregledov in oddaljeno spremljanje pacienta na domu s pogostim zajemanjem različnih zdravstvenih meritev.

Na spletnem portalu, namenjenemu zdravnikom, imajo ti možnost pregleda vseh svojih pacientov, njihovih podatkov in izpolnjenih zdravstvenih pregledov (opravljenih meritev in odgovorov na vprašanja). V tabelah so prikazani rezultati iz izpolnjenih vprašalnikov, lahko pa se vidi le rezultate določenega tipa meritve (npr. krvni sladkor, srčni utrip ...). Te meritve se lahko prikaže tudi na grafih, iz česar se lažje opazi spremembe in trende v podatkih. Zelo verjetno je, da zdravniki ob namestitvi platforme dobijo že nekaj pripravljenih zdravniških pregledov za osnovna bolezenska stanja, ki jih lahko potem prikrojijo po svojih potrebah oziroma dobijo dostop do testnega portala, kjer se lahko priučijo ustvarjanja pregledov in ostalih funkcionalno-

sti. Sicer pa imajo vsi zdravniki ene bolnišnice isti repozitorij pregledov, ki jih lahko pregledujejo ter si z njimi pomagajo. Preglede se lahko ureja (z ustvarjanjem novih verzij, pri čemer so starejše verzije še vedno dostopne) ali pa se ustvarja nove. Posameznemu pacientu se lahko določi enega ali več zdravstvenih pregledov, ki jih mora izpolnjevati. Lahko se naredi tudi skupine vprašalnikov, ki se jih potem določi celotnim skupinam pacientov z isto ali podobno boleznijo (npr. težave s srcem). Tipom meritev se lahko nastavlja meje, in sicer se jih določa za skupine pacientov ali pa vsakemu pacientu posamezno. Te meje se potem lahko uporablja pri zdravniških pregledih, kjer lahko iz gradnikov z določenimi mejami ustvarimo različne vejitve.

Za kreiranje zdravstvenega pregleda se uporablja interaktivni portal, ki omogoča drag & drop (primi in potegni) raznih gradnikov na delovno ploščo, s čimer potem naredimo ustrezen delovni tok. Pregled se torej izdela tako, da se na delovno ploščo dodaja razne gradnike. Začeti je treba s t. i. začetnim gradnikom (start) in končati s končnim (end). V pregled je potem mogoče dodati še slednje gradnike:

- *meritev* (npr. meritev srčnega utripa, pritiska, bolečine ...) – meritev je lahko pridobljena s povezane naprave ali pa je ročno vnešena;
- *vprašanje*, kjer pacient vpiše odgovor v obliki številke, teksta ali pa logične vrednosti (da/ne);
- *izbiro* med več že pripravljenimi možnostmi odgovorov;
- *zaporedje izbir* z že pripravljenimi možnostmi odgovorov (vsakemu odgovoru se določi številska vrednost, glede na pacientove odgovore se na koncu vsota vrednosti odgovorov uporabi za delitev na poti);
- *zakasnitev* v sekundah.

Gradnike je nato treba med seboj povezati s puščicami, s katerimi določimo zaporedje, v katerem si dani gradniki sledijo. Večina omogoča nadaljevanje le po eni poti, iz nekaterih gradnikov pa se lahko definira več poti.

Pri meritvi je mogoče glede na meje, ki jih določimo drugje na portalu, v in iz gradnika potegniti puščice. Prav tako peljeta iz vprašanja, na katerega odgovorimo z da/ne, dve poti. Tudi pri izbiri med več možnostmi je za vsak odgovor mogoče peljati drugo pot. Po kreiranju in dodeljevanju zdravstvenega pregleda določenemu bolniku se mu ta pojavi v mobilni aplikaciji, namenjeni pacientom. Aplikacijo je mogoče naložiti le na naprave z operacijskim sistemom Android. Z izpolnjevanjem vprašalnika začnemo s klikom nanj v aplikaciji. Po končanem reševanju se nam pojavi vprašanje, če želimo podatke poslati zdravniku. V aplikaciji lahko pacient tudi pregleduje podatke o svojih opravljenih meritvah ter sporočila, ki si jih je izmenjal z zdravnikom [18].

Medable

Večina aplikacij, ki jih Medable ponuja, je narejenih z drugačnim ciljem kot Opentelehealth in so tako bolj namenjene izvajanju različnih zdravstvenih raziskav na velikem številu ljudi z določeno boleznijo. Ne osredotočajo se torej na relacijo zdravnik – pacient, ampak se skuša z zbranimi podatki na velikem številu ljudi natančno preučiti bolezni in njihove znake ter najti rešitve za neko bolezensko stanje (npr. Alzheimerjeva bolezen). Tako so aplikacije bolj kot za osebne zdravnike ustvarjene za tiste, ki se ukvarjajo z raziskovanjem. Temeljijo na Applovem ogrodju ResearchKit [25], ki je namenjeno programiranju aplikacij, namenjenih medicinskim raziskavam. Ker je večini zdravnikom programiranje tuje, so se pri Medablu odločili, da bodo ustvarjanje takšnih aplikacij poenostavili in naredili bolj prijazno tudi neprogramerjem.

Trenutno ponujajo pet orodij (Axon, Synapse, Cortex, Cerebrum, Core). Raziskovalci preko portala Axon ustvarjajo vprašalnike, uporabniki pa si aplikacijo naložijo na mobilne telefone (tiste, ki uporabljajo operacijski sistem iOS ali Android) in preko nje rešujejo vprašalnike. Meritve je mogoče zajemati tudi z uporabo pametnih ur. Cortex nudi orodja za izdelavo aplikacij, pri čemer programerjem ni treba programirati zalednih sistemov, ampak

se posvetijo le uporabniškemu vmesniku. Na ta način lahko aplikacijo za reševanje vprašalnikov ustvarijo sami, pri čemer v kodi uporabljajo ustrezne klice na storitve REST, ki jih nudi Cortex. Ostale aplikacije, razen Synapse, ki je bila izdana v začetku junija in je namenjena oddaljenemu spremljanju pacientovih vitalnih znakov v živo, pa so bolj namenjene sami analizi pridobljenih podatkov.

Kreiranje zdravstvenih pregledov tu poteka na drugačen način kot pri Opentelehealthu, in sicer tabelarično preko vnosnih in izbirnih polj. Pregled (study) je sestavljen iz več nalog (task), ki jih sestavljajo koraki (step). Obstaja pet tipov nalog (npr. privolitev (consent), upravičenost (eligibility), vprašalnik (survey) ...). Z nalogo tako označimo neko zaključeno celoto korakov (npr. registracija, meritve in vprašanja, ki se tičejo delovanja srca). Posamezen korak lahko vsebuje eno ali več vprašanj, ki so lahko različnih tipov (tekstovno navodilo, meritev, vprašanje, izbira številske vrednosti na skali, določitev časa, zajem slike, določitev lokacije ...). Možno je tudi dodajanje novih tipov vprašanj (z ustvarjanjem posebnih objektov). Glede na tip koraka se potem določi korake, ki sledijo. Če iz koraka vodi več različnih poti, se z vejitvenimi pogoji (branching rule condition) določa, kateri koraki sledijo.

Podobnosti in razlike

Čeprav se v glavnem namena platform razlikujeta, lahko opravimo primerjavo ustvarjanja in izvajanja samih vprašalnikov. Oba portala v pregledu omogočata različna vnosna polja in zajem meritev. Paleta različnih vnosnih polj je pri Axonu precej širša, vendar pa Opentelehealth omogoča povezovanje večjega števila naprav in vsebuje večji nabor različnih vrst meritev že direktno v aplikaciji. Prav tako je pri obeh možno narediti različne vejitve. Aplikaciji se tu razlikujeta v načinu kreiranja vejitev in števila različnih poti, ki vodijo iz določenega koraka. Opentelehealth pri meritvah omogoča le pet različnih vejitev glede na prej določene meje, medtem ko jih je pri Axonu mogoče narediti večje število (in to za vsak korak posebej). Čeprav lahko

to po eni strani štejemo za pomanjkljivost Opentelehealtha, saj morda v nekaterih primerih potrebujemo večje število možnih nadaljevanj, pa je to zelo dobra stvar z vidika prilagajanja vprašalnikov posameznim pacientom ali skupinam pacientov. Pri ustvarjanju vprašalnika se le določi, kaj se zgodi, če izmerjena vrednost preseže različne meje, potem pa dejanske vrednosti teh meja določimo posebej za vsakega pacienta ali skupino pacientov (oziroma pustimo privzete vrednosti). Sicer pa omejitev z le petimi možnostmi vejitev ne velja pri vprašanjih z več pripravljenimi odgovori, saj tam meje za delitve na različne poti določamo direktno na delovni plošči. Obe aplikaciji tudi omogočata, da se pri deljenju na poti upošteva več pogojev (Axon omogoča dodajanje poljubnih pogojev, pri Opentelehealth pa je to možno le z uporabo gradnika zaporedja izbir z že pripravljenimi možnostmi odgovorov). Opentelehealth tudi omogoča dodajanje gradnika za zakasnitev, s katerim lahko npr. določimo, da mora pacient pred merjenjem krvnega pritiska nekaj časa počivati. Na zaslonu se takrat prikaže odštevalnik časa, pacient ne more nadaljevati z izvajanjem pregleda, dokler čas ne poteče. Axon takšne opcije nima. Vendar pa se pri Axonu lahko tako celotnemu pregledu kot tudi večini nalogam nastavi, kdaj se bodo izvedli. Lahko se izbere, da se pregled ali naloga izvede le enkrat, vsaki dve uri, določen dan, določen teden itd. Pri Opentelehealthu je mogoče čas izvedbe določiti le celotnemu pregledu, ne pa tudi posameznim gradnikom.

2.1.2 ZephyrLIFE™ Home Remote Patient Monitoring System [35] [36]

Platforma je namenjena ljudem s kroničnimi boleznimi in tistim, ki se rehabilitirajo po poškodbah ali operacijah. Aplikacija rešuje problema prevelikega števila sprejemov v bolnišnicah in zapravljenega časa, ki ga pacienti preživijo hospitalizirani. S tem tudi zmanjšuje pripadajoče stroške. Kolikor je mogoče razbrati iz tako omejenih virov, gre tu le za zajem vitalnih znakov in ne za opravljanje pregledov, ki jih kreira zdravnik.

Aplikacija podpira brezžično zajemanje in spremljanje pacientovih po-

datkov. Podatki se zajemajo simultano vsakih 15 minut, lahko se povezuje različne zunanje naprave, kot so naprave za merjenje teže, krvnega pritiska, krvnega sladkorja, telesne temperature. Glede na meje, ki jih nastavi zdravnik, se lahko prožijo različna opozorila. Uporabljata se 2 glavni napravi: *biopatch wireless device* in *healthub platform*. Biopatch se z elektrodami pritrdi na prsi, z njim se lahko zajema pulz, frekvenco dihanja, pozicijo (sedenje, ležanje), aktivnost (hoja, tek ...). Healthub platforma deluje na napravah z operacijskim sistemom Android in omogoča beleženje podatkov povsod, kjer je mogoče povezovanje v mobilno omrežje. Na napravi se prikazujejo podatki, ki jih zajema biopatch ali pa druge brezžične naprave. S portala, namenjenega zdravnikom, je mogoče za vsakega pacienta videti zgodovino meritev ter trende meritev na razni grafih.

2.1.3 Alayacare

Kot smo lahko razbrali iz zelo omejenih virov, ima platforma veliko različnih funkcionalnosti. Namenjena je shranjevanju podatkov o pregledih direktno v aplikaciji [19](s tem se zmanjšuje uporaba papirja, vsi podatki o pregledih so dostopni na enem mestu); urejanju, beleženju in lajšanju delovnih procesov medicinskih delavcev [6]; omogočanju dostopa do medicinskih podatkov tudi svojcem oz. drugim članom družine [1].

Platforma v prvi vrsti skrbi za beleženje in lajšanje delovnih procesov patronažnih sester, ki skrbijo za bolnike na domu. Poleg tega pa nudi tudi zajemanje zdravstvenih podatkov tem bolnikom. Del, ki zadeva zajemanje pacientovih vitalnih znakov na domu, omogoča izvajanje virtualnih obiskov preko telefona, reševanje vprašalnikov, izvajanje video konferenc in zajemanje vitalnih znakov. Možno je ustvarjati preglede, ki jih potem pri bolnikih na domu izvedejo medicinski delavci, poleg tega pa se lahko naredi tudi preglede, ki so namenjeni bolnikom samim, da jih ti samostojno opravljajo na domu. Omogočeno je povezovanje velikega števila naprav za zajemanje zdravstvenih podatkov. Preglede se lahko na enostaven način ustvarja in spreminja. Za kreiranje zdravstvenih pregledov se uporablja tabelaričen

način preko vnosnih in izbirnih polj (podobno, kot je to narejeno pri portalu Medable Axon) [2]. Omogočeno je dodajanje vprašanj različnih tipov, npr. vprašanje o počutju pacienta, zajem meritev krvnega pritiska, srčnega utripa ... Za zajete parametre se lahko določi meje, ob katerih se pojavljajo različna opozorila. Zelo verjetno je tudi podprto odločanje oz. dodajanje vejitev glede na opravljene meritve (v citiranem videoposnetku omenjeno kot *integrated decision support*). Aplikacijo za medicinske delavce in paciente lahko uporabljajo uporabniki iOS in Android naprav. Prenos je možen preko App Stora in Google Playa, vendar aplikacije ni mogoče kar tako uporabljati, ampak je potrebno pridobiti dostop do nje. Glavni portal, namenjen zdravnikom in administratorjem, je spletna aplikacija. Nudijo še družinski portal, preko katerega lahko do pacientovih podatkov dostopajo tudi njegovi družinski člani. *Alayalabs* aplikacija pa je namenjena analizi zdravstvenih podatkov in izvajanju strojnega učenja na njih.

2.1.4 Vivify health

Ponujajo več različnih produktov (Vivify Pathways Home, Vivify Pathways Go, Vivify Pathways Voice, Vivify Pathways Active) in API, ki omogoča programiranje novih aplikacij [32]. Home in Go sta namenjena spremljanju ljudi, ki so preboleli določeno bolezen ali opravili kakšno operacijo, kroničnim bolnikom, starostnikom, ženskam z rizično nosečnostjo [34] [33] ... Poleg namenske naprave (tabličnega računalnika) k produktoma spadajo še različne zunanje naprave za zajem zdravstvenih meritev. Sicer pa se aplikacijo lahko uporablja na različnih vrstah naprav, saj je ustvarjena z uporabo HTML5 ogrodja, kar pomeni, da se jo lahko odpira v različnih brskalnikih. Pacientom se lahko izmed več kot 75 pregledov, namenjenim spremljanju različnih bolezni, izbere primernega za njihovo zdravstveno stanje, ti pa ga potem izvajajo doma. Preglede je mogoče tudi urejati.

2.1.5 Tactio health

V ponudbi imajo več aplikacij [29]. RPM1000 Patient App je aplikacija za iOS in Android, ki omogoča beleženje vitalnih znakov, kot so teža, krvni tlak, utrip srca, saturacija krvi, telesna temperatura, kvaliteta spanja, počutje, število opravljenih korakov čez dan, fizična aktivnost ... Možno je ročno vnašanje podatkov, sicer pa se parametre pridobiva z raznih naprav, s katerimi se lahko mobilna naprava poveže. Aplikacijo lahko uporablja vsak, saj je možen prenos preko App stora in Google Playa. Z uporabo si lahko beležimo naše stanje in opazujemo naš napredek skozi čas. Aplikacija je tako sama po sebi namenjena samostojni uporabi, vendar se zajete podatke lahko uporablja tudi tako, da nam te spremlja naš zdravnik in npr. v primeru velikih odstopanj ustrezno ukrepa. Zdravnik v tem primeru uporablja aplikacijo RPM6000, ki mu omogoča spremljanje vseh pacientov in njihovih meritev. Na preprost in razločen način lahko vidi, kateremu pacientu je katera izmed vrednosti presegla dovoljene meje. Prav tako se lahko za vsakega pacienta vidi vse nazadnje zajete podatke, njegovo zgodovino za vsak tip meritev, pa tudi različne grafe. To zdravniku omogoča tudi kontroliranje bolnikov s kroničnimi obolenji. Velika pomanjkljivost aplikacije je ta, da ni mogoče ustvarjati zdravniških pregledov za različne vrste bolezni. RPM7000 je platforma s podprtim shranjevanjem podatkov v oblaku in je namenjena razvijalcem aplikacij.

2.1.6 Honeywell LifeCare

Ponujajo namensko napravo (Genesis DM) brez LCD ekrana in tablično aplikacijo (Genesis Touch [13]) za spremljanje pacientovih zdravstvenih podatkov na domu [14]. Naprava lahko pridobiva podatke s povezanih namenskih naprav. Zajeti je možno koncentracijo kisika v krvi, krvni tlak, težo in telesno temperaturo. Podatke se lahko tudi ročno vpisuje. Omogočeno je postavljanje nekaj vprašanj, ki pa so verjetno določena, že preden pacient dobi napravo k sebi domov. Tudi tu ne gre za izpolnjevanje vprašalnika oz. zdravniškega

pregleda, ampak pacient izbere, katere meritve želi opravljati oz., ali želi odgovarjati na vprašanja. Žal so viri, iz katerih smo pridobili zgornje podatke, precej zastareli.

2.1.7 Phillips eCare

Na spletu je zelo malo informacij o tej platformi. Se pa s spletne strani lahko razbere, da ponujajo aplikaciji eCareCompanion (za paciente) [20] in eCareCoordinator (za zdravnike) [21]. Preko aplikacije za paciente je mogoče povezovanje z raznimi napravami za zajem zdravstvenih podatkov (teža, krvni sladkor, razmerje kisika v krvi, krvni pritisk in pulz), možno je izpolnjevanje vprašalnikov, ki jih zdravniki ustvarijo preko eCareCoordinatorja in jih pošljejo pacientom. Primer specializirane uporabe je oddaljeno spremljanje bolnikov s kronično obstruktivno pljučno boleznijo, kjer ponujajo tudi različne naprave za merjenje kisika v krvi. Aplikacija za zdravnike pa tem omogoča pregled pacientov in njihovih meritev. Zbrani podatki se hranijo v oblaku, kjer jih je mogoče kasneje tudi analizirati. HealthSuite digitalna platforma je namenjena razvijalcem, da lahko z njeno pomočjo programirajo nove aplikacije, namenjene zdravstveni oskrbi [22].

2.2 Primerjava vseh obravnavanih rešitev

Na koncu smo vse analizirane aplikacije med seboj primerjali ter definirali njihove prednosti in slabosti. Iz primerjave smo nato lahko določili funkcionalnosti, ki jih naš portal za ustvarjanje zdravniških pregledov nujno potrebuje, da bo lahko konkuriral ostalim, in tiste, ki bi lahko pomenile veliko dodano vrednost in s tem prednost pred drugimi produkti. Na žalost je pri večini pregledanih ponudnikov težko pridobiti vzorčne aplikacije, v primeru ko jih ne zahteva katera izmed zdravstvenih ustanov. Prav tako na spletu delijo zelo malo podatkov o načinu delovanja njihovih aplikacij, njihovi arhitekturi in drugih tehničnih podrobnostih ali pa so najdene informacije stare že več let.

Najprej smo primerjali platforme glede na namen oz. težave, ki jih rešujejo. Ugotovili smo, da je cilj večine preučenih aplikacij zdravnikovo spremljanje vitalnih znakov pacienta na domu, saj se s tem ukvarjajo vse, razen Medabla, ki pa stopa na to področje z novo aplikacijo Synapse. Manj kot polovica se ukvarja s pridobivanjem podatkov o vitalnih znakih z namenom opravljanja širših raziskav nad temi podatki. Le Alayacare pa ponuja tudi beleženje obiskov in opravil patronažnih sester, ki obiskujejo paciente na domu. Razen Medabla in Tactio healtha vsi omogočajo vzpostavljanje video klica med pacienti in zdravniki, pri ZephyrLIFU pa glede tega ni mogoče najti nikakršnih podatkov, tako da predvidevamo, da ta funkcionalnost ni omogočena. Večina ponudnikov kot glavno napravo uporablja različne vrste mobilnih naprav in tabličnih računalnikov, pri Honeywellu pa so razvili tudi posebno napravo, ki ne uporablja LCD zaslona. Vse aplikacije delujejo na operacijskem sistemu Android, z izjemo Vivifya in Phillips eCara, kjer nam ni uspelo pridobiti informacij o operacijskem sistemu, na katerem njihova aplikacija teče. Aplikacijo za iOS pa ponujajo Medable, Alayacare in Tactio. Opentelehealth in Vivify pacientom nudita tudi vnos podatkov preko spletne aplikacije, vendar se v tem primeru pri Opentelehealthu ne da zajemati meritev iz povezanih naprav, ampak je mogoč le ročen vnos. Izmed vseh podpira Medable najmanj različnih brezžičnih naprav za zajem zdravstvenih podatkov pacientov. Tam se namreč podatki v večini pridobijo samo preko mobilnega telefona, možno pa je povezovanje z nekaterimi pametnimi urami. Je pa večina ponudnikov omejena le na peščico naprav določenih ponudnikov. Izjemi sta Alayacare, ki omogoča povezovanje preko 200 različnih zunanjih naprav [1], in Tactio, ki prav tako podpira precejšnje število naprav različnih znamk [30]. Opentelehealth, Alayacare, Vivify in Phillips eCare uporabljajo svoje, zasebne oblačne storitve, pri ostalih platformah pa o tem ni mogoče najti nobene informacije.

Na koncu smo se osredotočili še na analizo in primerjavo tistega dela platform, s katerim se bomo ukvarjali tudi v naši diplomski nalogi – z možnostjo ustvarjanja zdravstvenih pregledov za spremljanje pacientov na domu in nji-

hovega izvjanja. Prišli smo do ugotovitev, da izvajanje zdravstvenih pregledov oz. izpolnjevanje vprašalnikov omogoča dobra polovica platform. ZephyrLIFE, Tactio in Honeywell namreč omogočajo le zajem vsakega zdravstvenega podatka posebej in ne specialnih pregledov z različnimi tipi vprašanj in navodili za zajem meritev, ki bi jih izpolnjevali pacienti. Za samo ustvarjanje vprašalnikov obstoječe platforme uporabljajo dva načina, in sicer pri Opentelehealth uporabljajo interaktivni portal, kjer je mogoč drag & drop različnih gradnikov na delovno ploščo, pri Medable in Alayacare pa tabelarični način z izpolnjevanjem vnosnih in izbirnih polj. Informacij o načinu ustvarjanja pregledov pri ZephyrLIFE, Vivify in Phillips eCare žal nismo uspeli najti.

2.3 Ustvarjanje zdravniških pregledov

Oba pristopa za kreiranje zdravniških pregledov imata svoje prednosti in slabosti. Žal nam ni bilo omogočeno preizkusiti aplikacije Axon, da bi lahko dejansko primerjali, kako poteka kreiranje vprašalnika z uporabo tabelaričnega načina in koliko časa za to porabimo. Smo pa zato dodobra stestirali aplikacijo Opentelehealth in s tem interaktivno možnost modeliranja.

Ustvarili smo več različno obsežnih in kompleksnih vprašalnikov in ugotovili, da je interaktivni način zelo intuitiven, saj je mogoče razločno videti, kako so gradniki med seboj povezani in po kateri povezavi se iz nekega koraka gre, glede na vrednost pacientove meritve ali odgovora. V primeru kompleksnejših pregledov pa postane premikanje gradnikov po plošči zamudno, saj je treba vsak gradnik premakniti na svoje mesto. Pozicijo je ob nastajanju novih povezav treba mnogokrat popravljati. Tudi samo sosledje korakov zelo hitro ni več tako enostavno razvidno, saj postane delovna plošča natrpana in težko obvladljiva. Za določanje lastnosti posameznim gradnikom (npr. naslova in opisa gradniku za vprašanje) je pri obeh načinih ponavadi potrebno enako število akcij, le da pri tabelaričnem načinu gradnikov ni treba pozicionirati. Kar pri tabelaričnem načinu vzame nekaj več časa in je manj

samoumevno za uporabo, je dodajanje vejitev oz. določanje povezav med koraki. Z grafičnega prikaza poti lahko namreč zelo hitro razberemo vejitve med gradniki. To pomanjkljivost tabelaričnega modeliranja lahko deloma odpravimo tako, da aplikaciji dodamo funkcionalnost, ki omogoča grafični prikaz korakov in povezav, ki jih je uporabnik dodal. Dodana bi lahko bila tudi možnost predogleda, ki bi omogočala zdravniku, da med ustvarjanjem pregleda preveri, kako bo ta izgledal na pacientovi napravi.

Glede na vse omenjene vidike ni noben od pristopov izrecno boljši od drugega. Lahko rečemo, da je interaktiven način ustvarjanja pregledov bolj intuitiven in se ga je možno hitreje naučiti, veščemu uporabniku pa tudi tabelarična izdelava pregleda povsem ustreza in je v primeru velikega števila gradnikov lahko še hitrejša. Kljub temu, da nismo mogli dejansko preizkusiti tabelaričnega pristopa, lahko predpostavljamo, da tudi uporaba tabelaričnega načina z uporabniškega stališča nima tako velikih pomanjkljivosti, ta pristop so namreč uporabili tako pri Alayacaru kot tudi pri Medablu. Sploh ravnanje Medabla nam lahko služi kot dober zgled, saj je produkt Axon zelo nov in je na voljo šele od konca septembra 2016.

Naši platformi bi veliko dodano vrednost dodal tudi repozitorij delovnih tokov, kamor bi zdravniki lahko shranjevali svoje preglede in jih delili z zdravniki, ki delajo v drugih institucijah. Česa takega namreč ne omogoča nobena izmed pregledanih platform. Dobro bi bilo tudi, da bi se preglede ocenjevalo glede na to, kako služijo svojemu namenu, kar bi omogočalo zdravnikom lažje odločanje pri izbiri ustreznih vprašalnikov.

Poglavje 3

Načrtovanje zdravniških pregledov

Po temeljitem razmisleku in dognanjih iz prejšnjega poglavja smo se odločili za ustvarjanje zdravniških pregledov uporabiti tabelarični način, pri čemer smo skušali z implementacijo dodatnih funkcionalnosti, kot je npr. grafični prikaz, odpraviti oziroma omiliti njegove pomanjkljivosti.

Zdravniški pregled je sestavljen iz več korakov, ki si med seboj sledijo v določenem zaporedju. Koraki so lahko različnih tipov, v osnovi pa jih najprej delimo na dva tipa – *vprašanje* in *meritev*. Tu je še poseben tretji tip, ki predstavlja množico več vprašanj in/ali meritev.

Za zdravniški pregled je značilnih nekaj lastnosti, ki mu jih je potrebno ob ustvarjanju definirati. Vsak pregled ima svoj naslov in opis, določi pa se mu tudi vrsto kronične bolezni, za katero bo namenjen. Dalje je tu še podatek o aktivaciji pregleda. V kolikor pregled še ni aktiviran, to pomeni, da ga zdravnik pacientom še ne more dodeliti in ga ti ne morejo reševati. Lahko pa se takrat pregled ureja in spreminja. Urejanje korakov pregleda pa ni več mogoče, ko pride do aktivacije, saj bi v nasprotnem primeru prihajalo do nekonsistentnih podatkov, pridobljenih od pacientov. Za nekatere paciente bi namreč bili shranjeni podatki o nespremenjenih pregledih, za druge pa o spremenjenih. To bi povzročalo težave pri prikazovanju podatkov in pri

primerjavi vrednosti skozi čas. Ko je torej pregled enkrat aktiviran, ga ni več mogoče spreminjati, lahko pa se naredi novo verzijo, kjer se ga tako popravi ter dopolni.

3.1 Koraki

Zdravniški pregled je sestavljen iz več korakov. Korak predstavljajo različne meritve (npr. meritve krvnega tlaka, srčnega utripa, teže ...) in vprašanja ali navodila, ki jih zdravnik zastavlja pacientu. Koraki si sledijo v vrstnem redu, ki ga določi zdravnik.

3.1.1 Vprašanje

Prvemu tipu koraka smo nadeli oznako *vprašanje*, saj v večini primerov predstavlja prav to. Obstaja več tipov vprašanj, pri čemer je za vsakega izmed njih značilno to, da mu nastavimo *naslov* in *opis/vprašanje*. Za osnovne tipe smo določili:

- *vnosno polje* – predstavlja vrsto vprašanja, kjer se od uporabnika pričakuje, da svoj odgovor vpiše v vnosno polje. Preprost primer takega vprašanja bi bil: „Z nekaj besedami opišite, kaj ste danes jedli.“ Odgovor je lahko niz znakov ali število, vendar se različnih tipov odgovorov ne ločuje med seboj, ampak se vse tretira kot niz;

- *enojna izbira* – tu se uporabniku ponudi več možnih odgovorov, izbere pa lahko le enega izmed njih. Zdravnik pri dodajanju koraka te vrste določi možne izbire. Primer bi bilo vprašanje „Pred koliko urami ste imeli zadnji obrok?“, možni odgovori pa: „Pred 1 uro“, „Pred 1-2 urama“, „Pred 2-3 urami“, „Pred več kot tremi urami“;

- *večkratna izbira* – tudi tu ima uporabnik možnost izbirati med ponujenimi odgovori, vendar jih lahko izbere tudi več, ne le enega;

- *navodilo/sporočilo* – tip vprašanja, ki se od ostalih razlikuje po tem, da od uporabnika ne pričakuje odgovora, ampak se takrat le prikaže določeno sporočilo, npr. naj pacient pred nadaljevanjem izvajanja pregleda počiva nekaj minut, saj si bo pri nasledenjem koraku moral izmeriti krvni pritisk.

Opisani tipi vprašanj so osnovni, vendar lahko z njimi zajamemo veliko večino različnih vrst podatkov, ki jih potrebujemo. Da bi lahko zdravniki bolj učinkovito dodajali korake in bi bilo opravljanje pregledov bolnikom bolj prijazno, se lahko kasneje doda še druge tipe, kot so naprimer možnost številske izbire z uporabo drsnika, izbira datuma s koledarja, zajem slike, lokacije, ipd. Zaradi lažjega primerjanja vrednosti in zmanjšanja verjetnosti za napake pri vnosih bi bilo dobro vnosno polje nadgraditi tako, da se mu lahko določi še, ali naj bo pacientov vnos številski ali znakovni.

3.1.2 Meritev

Drugi tip korakov se uporablja za označevanje korakov, pri katerih se zgodi zajem določenega tipa pacientovega vitalnega znaka. Zaenkrat so bili dodani slednji tipi meritev:

- *meritev krvnega tlaka*
- *meritev srčnega utripa*
- *meritev telesne teže*
- *meritev sladkorja v krvi*
- *meritev saturacije kisika v krvi*
- *meritev telesne temperature*

Za vse tipe, razen za meritev krvnega tlaka, velja, da se pri koraku izvede meritev enega parametra vitalnega znaka. Pri merjenju krvnega tlaka

pride namreč do zajema treh parametrov – sistoličnega, diastoličnega in povprečnega krvnega pritiska. Zaradi tega in pa tudi zato, da se lahko v prihodnje doda še morebitne druge tipe meritev, ki so predstavljeni z več vitalnimi znaki, je ena meritev predstavljena z enim ali več parametri. Razlika med vprašanji in meritvami je tudi v tem, da pri meritvah ni mogoče nastavljati nobenih lastnosti, kot sta pri vprašanjih naslov in opis, ampak ustvarjalec pregleda le izbere tip meritve, ki se bo opravila na določenem koraku. Ker torej meritve ne omogočajo spreminjanja, ampak jih lahko določa in spreminja le programer, sta bila k njim dodana še dva tipa, katerih namen je sicer le programerski, in predstavljata začetek ter konec zdravniškega pregleda. Uporablja se ju le za lažje prikazovanje pregleda in določanje vejitev.

3.2 Vejitve

Da lahko zdravnik ustvarja kompleksnejše zdravniške preglede, ki omogočajo čim boljše posnemanje pregledov v živo, velja, da ni nujno, da se koraki izvajajo eden za drugim, v vrstnem redu, v katerem so bili narejeni. Potrebno je namreč omogočiti možnost vejitev, kar pomeni, da se lahko pacient pri opravljanju pregleda iz nekega koraka pomakne v več različnih, glede na vrednost opravljene meritve ali vnesenega odgovora. Zato mora imeti zdravnik možnost določanja vejitev za vsakega izmed korakov. Najprej se določi, kateri korak je naslednji, nato pa se še definira pravila, ki morajo biti izpolnjena, da se bo izvedel premik. Za preskok v korak se lahko določi enega ali več pogojev, med katerimi lahko velja konjuktivna ali disjuktivna odvisnost. Lahko se naprimer določi, da se, v kolikor je sistolični tlak višji od 139 mmHg in diastolični tlak višji od 89 mmHg, zgodi premik v korak, kjer se pacientu prikaže sporočilo, naj vzame zdravila za zmanjšanje krvnega pritiska. Če pa ta pogoja nista izpolnjena, se pregled zaključi.

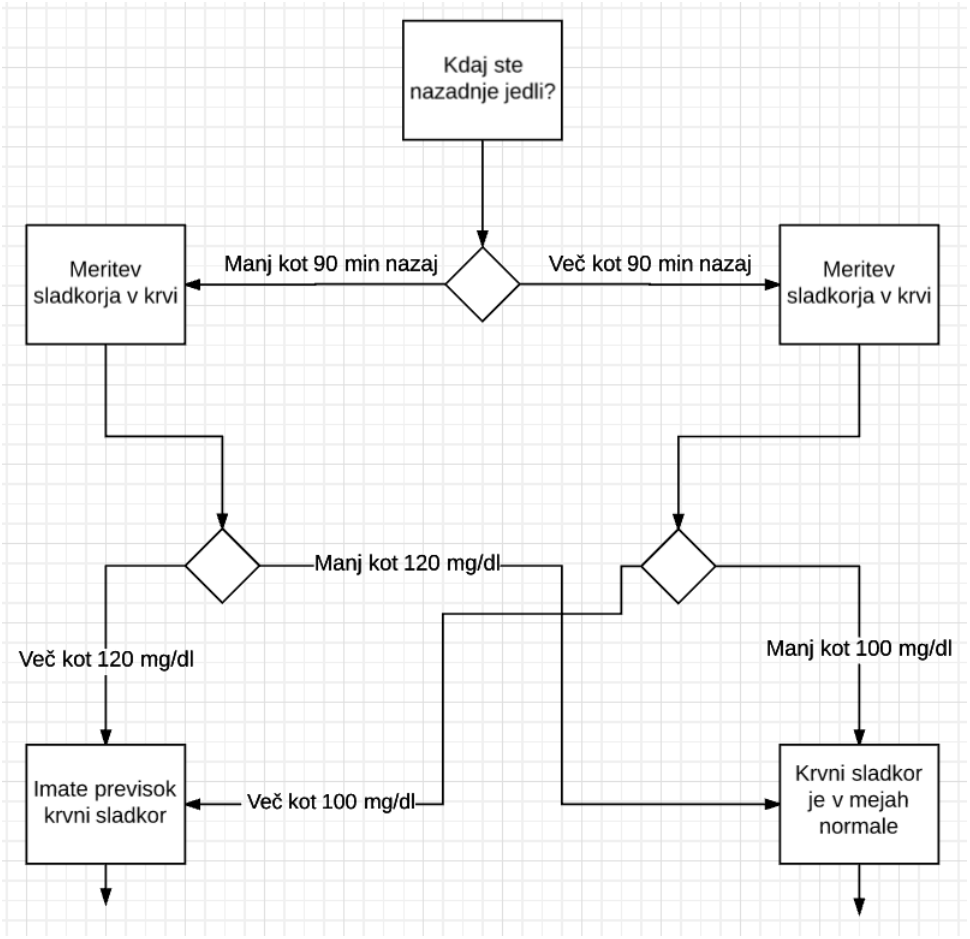
Pri vejitvah bi lahko bilo omogočeno, da se iz nekega koraka zgodi premik tudi na prejšnje korake. Tako bi zdravniški pregled vseboval zanke. Vendar pride v tem primeru do vprašanja, kaj narediti z vrednostmi, pridobljenimi

v prvem obhodu zanke. Če bi bil pregled narejen tako, da se nova iteracija zgodi le v primeru, da je meritev na nek način napačna (izven „normalnih“ vrednosti), bi bilo to uporabno. Vendar pa tega ne moremo zagotoviti, saj se zdravnikov ne da omejiti, da bodo zanke naredili le v takih primerih. Lahko bi sicer hranili podatke tako iz prve kot tudi druge iteracije, vendar nastanejo komplikacije s poznejšim prikazovanjem podatkov. V nekaterih primerih bi bilo tudi težko določiti, katera izmed meritev je bolj ustrezna. Hranjenje podatkov le iz druge iteracije, ko ne vemo, ali je vrednost prve relevantna, pa se tudi zdi zgrešeno, saj gredo tako meritve iz prvega obhoda v nič. Problem je tudi v tem, da postanejo z uporabo zank pregledi kompleksnejši, kar lahko pri ustvarjanju hitreje privede do napak. Zato je bila arhitektura aplikacije zasnovana tako, da zanke podpira, ampak so bile te nato naknadno onemogočene. Predvideva se, da bodo zdravniški pregledi narejeni tako, da se v primeru morebitnih napačnih podatkov raje še enkrat izvede celoten pregled.

3.2.1 Množica meritev in vprašanj

Na tem mestu lahko opišemo še tretji tip korakov, ki ga sestavlja več različnih meritev in/ali vprašanj. Potreben je zato, da se lahko v nekem koraku naredi vejitve z več pogoji, ki se nanašajo na različne meritve in vprašanja. S tem se lahko precej poenostavi ustvarjanje vprašalnikov, saj ni potrebno ponavljanje korakov. Če bi želeli od pacienta dobiti podatek o tem, koliko časa nazaj je jedel in kakšen je njegov krvni sladkor, ter nato glede na obe informaciji določiti njegovo stanje, bi morali definirati korake na način, kot je prikazan na sliki 3.1. Težava, ki se tu pojavi, je ta, da je treba nekatere korake, v tem primeru meritev krvnega sladkorja, definirati dvakrat. To lahko postane zamudno, predvsem pa se s tem manjša preglednost seznama korakov.

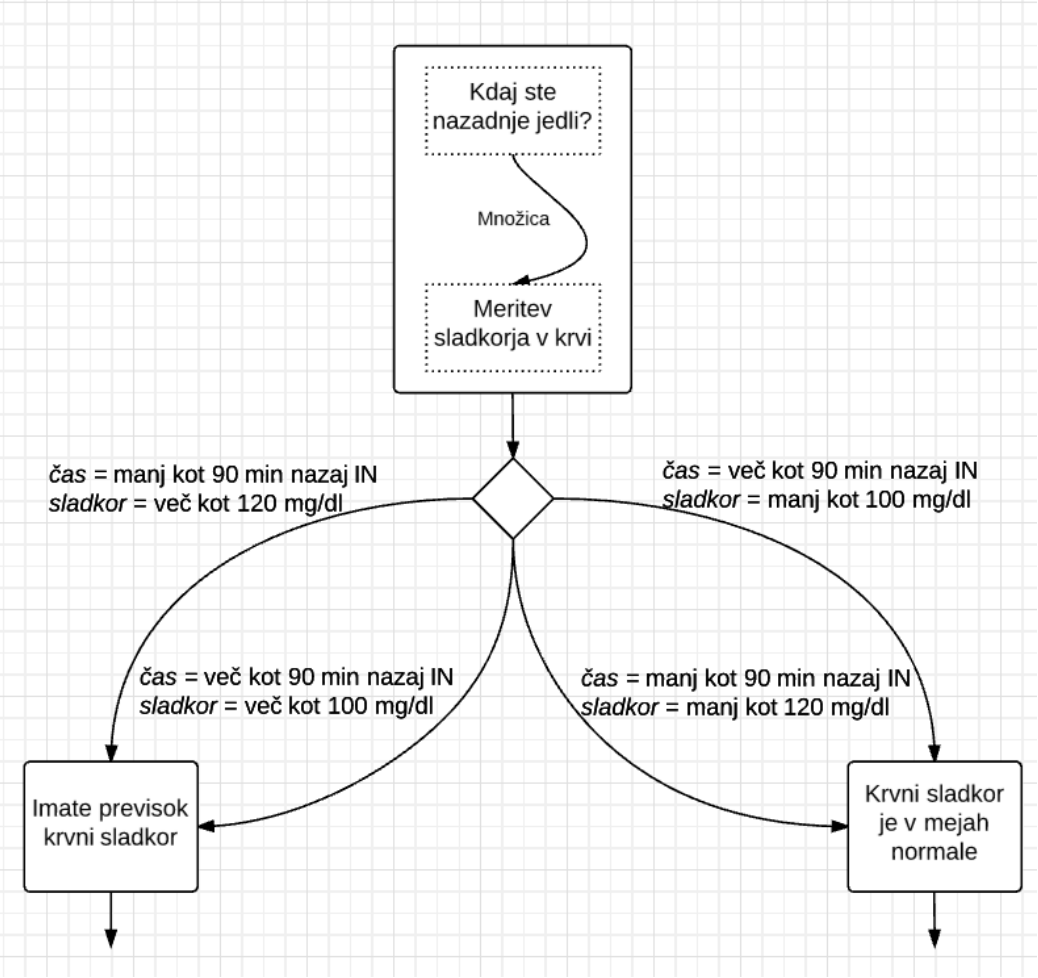
Težavo bi lahko v nekaterih primerih rešili tako, da bi bilo v vsakem koraku omogočeno dodajanje vejitev s pogoji, ki se nanašajo na vse meritve in vprašanja, ki jih je pacient opravil do tega trenutka. Vendar to ni vedno mogoče, saj se lahko do nekega koraka pride po več različnih poteh, zato



Slika 3.1: Primer zaporedja vprašanj in meritev brez uporabe tipa koraka, ki vsebuje množico vprašanj in meritev.

ni nujno, da bodo v nekem koraku dostopni vsi podatki, glede na katere se bi potem izvajalo primerjave. Na ta način bi hitro prišlo do napak in nedefiniranih stanj, ko bi se v nekem koraku naredila primerjava s podatki, ki niso bili vneseni. Zato je najboljša rešitev te zagate vpeljava posebnega tipa koraka, ki lahko vsebuje več meritev in/ali vprašanj. S tem se namreč določi, da mora pacient opraviti vsako izmed meritev tega koraka in odgovoriti na vsa vprašanja na tem mestu, do samih vejitev pa potem pride šele na koncu, ko so vsi potrebni podatki zagotovo že na voljo. Na sliki 3.2 je viden prej

omenjen primer z uporabo tretjega tipa koraka.



Slika 3.2: Primer zaporedja vprašanj in meritev z uporabo tipa koraka, ki vsebuje množico vprašanj in meritev.

Poglavje 4

Opis in predstavitev razvite spletne aplikacije

Za modeliranje zdravniških pregledov smo razvili spletno aplikacijo. V njej lahko zdravniki ustvarjajo nove preglede, jih spreminjajo ter dodajajo nove verzije. Ti pregledi bi se nato dodeljevali pacientom, ki jih bi na domu reševali na namenskih napravah. Veliko pozornost smo namenili temu, da se pregledi izmenjujejo v formatu, ki je čim bolj pregleden in enostaven. To potem omogoča lažje razčlenjevanje delovnih tokov na mestu, ki skrbi za prikaz pregledov pacientom. Pri ustvarjanju formata pregledov smo se skušali držati standarda FHIR, vendar smo naleteli na prevelike razlike v načinu predstavitve zdravniških pregledov, zaradi katerih smo to zamisel opustili. Namesto tega smo implementirali funkcionalnost, ki omogoča pretvorbo in shranjevanje pregleda FHIR v našo obliko.

Ker pa bi bilo programiranje aplikacije z vsemi podrobnostmi preobseženo za to diplomsko delo, smo del aplikacije, namenjen upravljanju s pacienti, izpustili. Dodajanja pregledov posameznim pacientom tako nismo implementirali, ampak smo naredili le prototipno spletno aplikacijo za paciente, ki zna ustrezno prikazovati zdravniške preglede, shraniti vnešene podatke in jih poslati strežniku.

Pri pregledu področja smo prišli do spoznanj, da je lahko medicinsko

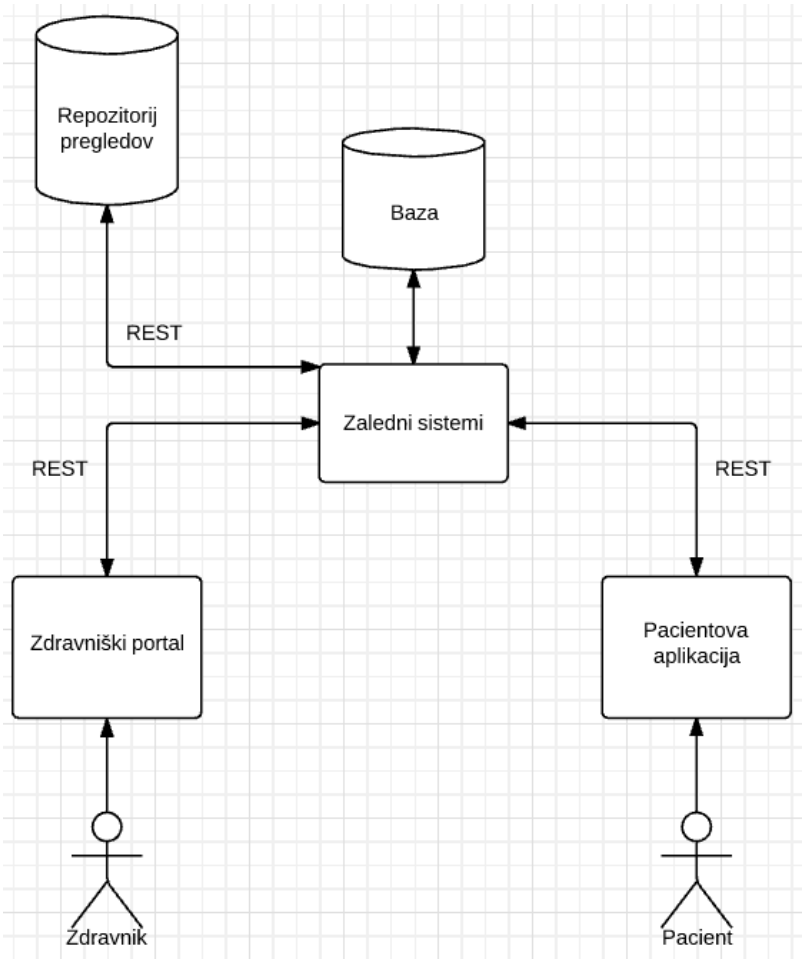
znanje še vedno velikokrat zelo izolirano in da zdravniki po svetu še vedno nimajo definiranih najboljših pristopov za zdravljenje oz. lajšanje kroničnih bolezni, ki bi jim lahko enostavno sledili. Veliko je k rešitvi tega problema sicer pripomoglo vpeljevanje kliničnih poti, vendar bi lahko spletna rešitev, ki bi omogočala, da si načine zdravljenja različnih kroničnih bolezni lahko deli zdravniško osebje povsod, kjer je omogočen dostop do spleta, to težavo povsem odpravila. Specialisti, ki veliko časa in truda namenijo preučevanju točno določene kronične bolezni, lahko namreč ustvarijo zdravniški pregled, ki omogoča pridobitev ključnih informacij, s katerimi se lahko spremljata stanje in napredovanje bolezni. S tem je mogoče bolezen lažje kontrolirati in zdraviti. Zato smo prišli do ideje, da bi k implementaciji naše rešitve dodali še globalni repozitorij, ki bi omogočal deljenje zdravniških pregledov zdravnikom različnih oddelkov in ustanov. Uporabniki od drugod si lahko preglede prenesejo v svojo aplikacijo in jih uporabljajo. Nekdo, ki te bolezni ne pozna dobro, si zato pregled raje prenese iz repozitorija, kot da bi ga ustvarjal sam. Ker smo pri analizi obstoječih rešitev ugotovili, da nobena izmed njih ne implementira tovrstne funkcionalnosti, je to tudi velika dodana vrednost naše aplikacije.

4.1 Arhitektura aplikacije

Pri načrtovanju arhitekture aplikacije smo se držali vzorca MVC, ki spletno aplikacijo razdeli na tri dele, in sicer na model, pogled in kontroler. Model predstavlja podatke, ki se prikažejo v pogledu. Kontroler pa predstavlja vmesno plast med modelom in pogledom, saj skrbi za to, kateri podatki se prikažejo, obenem pa glede na interakcijo uporabnika spreminja podatke v modelu.

Sicer pa rešitev temelji na arhitekturi REST, celotno strukturo lahko predstavimo na način, viden na sliki 4.1.

Aplikacija je sestavljena iz PostgreSQL podatkovne baze, kamor se shranjujejo tako zdravniški pogledi, kot tudi rezultati meritev in odgovorov, ki



Slika 4.1: Arhitektura aplikacije

jih opravijo pacienti. S podatkovno bazo komunicirajo zaledni sistemi, ki so napisani v programskem jeziku Node.js. Ti skrbijo tudi za komunikacijo z repozitorijem pregledov. Zaledni sistemi implementirajo storitve REST, na katere izvajata klice uporabniška vmesnika za zdravnike in paciente. Na voljo so tudi viri, preko katerih je mogoče spremeniti preglede, predstavljene po standardu FHIR, v našo obliko. Za programiranje uporabniških vmesnikov je bila uporabljena tehnologija Angular 4.

4.2 Tehnologije, uporabljene pri razvoju

Pri razvoju naše rešitve smo uporabili naslednje tehnologije:

- **PostgreSQL** – odprtokodni sistem za upravljanje s podatkovnimi bazami [23]. Sistem je objektno orientiran, odlikujeta ga zanesljivost in razširljivost. Podpira transakcije, pri čemer so transakcijam zagotavljene lastnosti ACID.

- **Node.js** – odprtokodno okolje, namenjeno izvajanju JavaScript kode na strežniku [17]. Odlikuje ga asinhron dizajn, ki omogoča ustvarjanje aplikacij, za katere je značilna učinkovita raba strežniških virov. To pa ima za posledico zelo dobro skaliranje. Ker omogoča hiter razvoj, smo ga uporabili za razvoj zalednih sistemov. Poslužili smo se še dveh Node.js ogrodij, in sicer ogrodja *Express*, s katerim smo ustvarili dostopne točke REST in implementirali logiko za odziv na HTTP zahtevke, ter *Sequeliza*, ki je namenjen povezovanju s podatkovno bazo in služi kot ogrodje za ORM. Node.js podpira pisanje programske kode v JavaScriptu in TypeScriptu – pri naši aplikaciji smo se odločili za slednjega.

- **Angular 4** – odprtokodna platforma za razvoj spletnih aplikacij, ki temelji na TypeScriptu in s katero je bil sprogramiran uporabniški vmesnik naše rešitve [3]. Začetki Angularja segajo v leto 2010, ko je bila izdana prva verzija AngularJs. Leta 2016 je bil izdan Angular 2, ki je na novo spisano ogrodje in nima s prejšnjo verzijo nič skupnega. Angular 4 pa je naslednik Angularja 2 in je z njim tudi kompatibilen. Pri progamiranju smo si pomagali z vmesnikom v ukazni vrstici (*Angular CLI*), ki omogoča enostavno upravljanje z Angular projektom. Za oblikovanje uporabniškega vmesnika so bili uporabljeni HTML, CSS in ogrodje *Bootstrap* [5]. To nudi uporabo predlogov komponent uporabniških vmesnikov (npr. gumbe, forme ...) ter omogoča določanje njihove postavitve. Uporabili smo še nekaj drugih knjižnic, s katerimi smo si olajšali delo (npr. knjižnici *d3.js* in *c3.js* za prikazovanje grafov).

- **GIT** – odprtokodni sistem, namenjen verzioniranju programske kode [9]. Čeprav se glavna prednost uporabe kaže pri projektih, na katerih sodeluje več ljudi, saj omogoča dobro koordinacijo dela na datotekah s programsko kodo, smo ga pri naši rešitvi uporabili predvsem z namenom verzioniranja in dobrega pregleda nad dodanimi funkcionalnostmi in opravljenimi spremembami. GIT smo povezali z *Bitbucketom* [4], kjer smo našemu projektu ustvarili repozitorij.

4.3 Podatkovni model

Implementacijo spletne aplikacije smo začeli z definicijo konceptualnega modela podatkovne baze, saj je to primer dobre prakse, ki se ga ponavadi uporablja pri razvoju aplikacij. Tako se namreč kasneje lažje določi vso potrebno logiko, ki jo je treba implementirati. Seveda se je marsikatera podrobnost podatkovnega modela tekom programiranja še spremenila, a osnova je ostala ista. Cilj je bil, da se lahko v podatkovno bazo shrani čim več različnih tipov zdravniških pregledov ter da relacije v bazi ne nudijo preveč omejitev. To namreč kasneje omogoča večjo prilagodljivost in lažje dodajanje različnih razširitev tekom razvoja aplikacije pa tudi kasneje. Sicer pa smo konceptualni model podatkovne baze določili na podlagi dognanj, ki smo jih predstavili v poglavju 3.

4.3.1 Tabele v podatkovni bazi

Končni model podatkovne baze je prikazan na sliki 4.2, sestavljajo pa ga tabele, predstavljene v naslednjih razdelkih.

ProcessDefinition, ProcessVersion in ChronicDisease

V tabeli *ProcessDefinition* so shranjeni osnovni podatki o zdravniškem pregledu. To so naslov, opis, datum kreiranja, datum zadnje spremembe ter podatek, ki pove, če takšen pregled obstaja tudi na globalnem repozitoriju

in če je bil tja naložen ali z njega prenešen. Opis je namenjen temu, da si z njim pomagajo zdravniki in tako dobijo predstavo o pregledu. Zato je dodan še opis za paciente, ki se tem prikaže na napravi pri izpolnjevanju. Naslov, opis in opis za paciente je mogoče kasneje še vedno spreminjati. Tabela je povezana z entiteto *ChronicDisease*, ki pove, na katero kronično bolezen se nanaša pregled.

Procesna definicija ima lahko eno ali več verzij; informacije o teh se hranijo v *ProcessVersion*. Shranjen je podatek o verziji, o tem, če je verzija aktivna, oziroma če je bila do sedaj že kdaj aktivirana, datum in čas stvaritve ter morebiten podatek o prisotnosti verzije na repozitoriju. Dva podatka o aktivaciji sta potrebna, da se lahko onemogoči urejanje procesne verzije po tem, ko je bila ta aktivirana, in tudi v primeru, ko je bila mogoče spremenjena nazaj v neaktivno. Ko je verzija enkrat ponovno neaktivna, je namreč težko vedeti, ali je v aktivnem času morda kateri izmed pacientov že opravil ta pregled. Poudariti je treba, da se ob aktivaciji onemogoči urejanje procesne verzije in vseh tabel, ki so vezane nanjo.

Step

Procesno verzijo sestavljajo koraki, ki so definirani v tabeli *Step*. Model bi sicer lahko naredili tako, da bi bila procesna verzija povezana direktno z meritvami ter vprašanji in bi to tabelo izpustili. Vendar to ni mogoče zaradi dveh razlogov. Prvi je ta, da bi se potem pojavljale nekonsistentne vrednosti v tabelah, ki hranijo podatke o vejitvah. Nekatera vprašanja bi bila namreč vezana na meritve, druga na vprašanja in obratno. Tabela *SequenceRule* bi tako potrebovala dva stolpca – enega za povezavo na tabelo meritev, drugega pa za povezavo na vprašanja. To bi pomenilo, da bi bilo v teh stolpcih veliko *null* vrednosti, zato bi bilo tudi v tem primeru najbolje narediti dodatno tabelo, ki bi jo entiteti meritev in vprašanj razširjali. Drugi razlog pa je ta, da zdravniški pregled omogoča vrsto koraka, ki predstavlja množico meritev in vprašanj. To pa se težko predstavi na drugačen način od omenjenega. *Step* tako nosi podatka o tipu koraka (vprašanje, meritev, množica), ki je le infor-

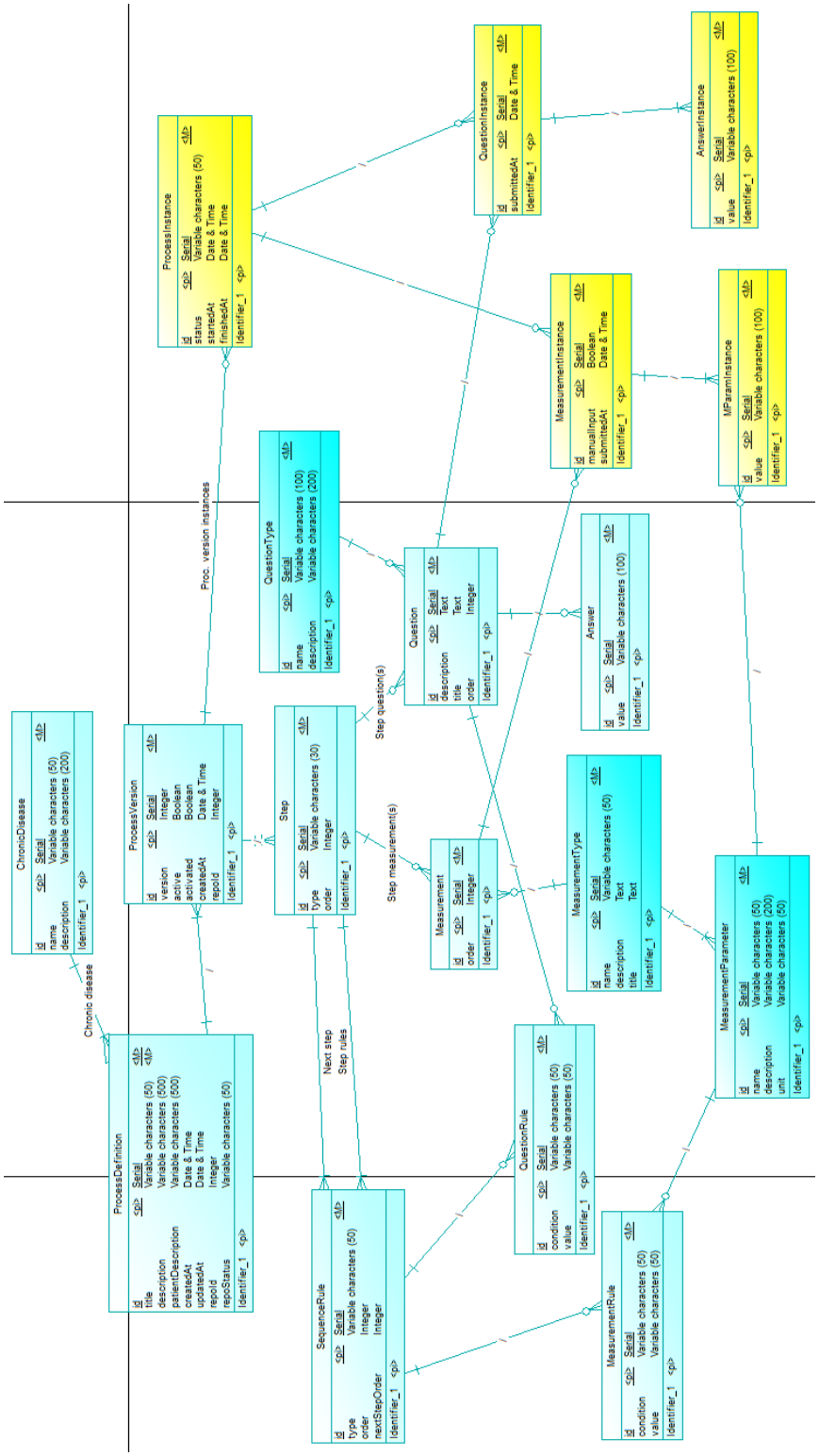
mativne narave in omogoča lažje poizvedbe po drugih tabelah in o vrstnem redu koraka. Tabela je vezana na *Measurement*, *Question* in *SequenceRule*.

Measurement, MeasurementType, MeasurementParameter

Korak lahko vsebuje eno ali več meritev, ki so shranjene v tabeli *Measurement*. Entiteta hrani le podatek o vrstnem redu meritve na tem koraku, vse ostale informacije o meritvi (naslov, opis ter oznaka meritve, na katero se sklicujemo v programski kodi) so na voljo v tabeli *MeasurementType*. V tej tabeli sta npr. kratek naslov in opis meritve krvnega pritiska. Dejanski parametri, ki se morajo zajeti, pa so shranjeni v tabeli *MeasurementParameter*; ta poleg oznake, ki se jo uporablja v kodi, vsebuje še ime parametra in enoto. Za omenjeni primer so tu trije vnosi – za sistolični, diastolični in povprečni krvni tlak. Tabeli *MeasurementType* in *MeasurementParameter* sta na prikazu modela pobarvani s temnejšo barvo; s tem smo želeli označiti, da se od ostalih tabel razlikujeta po tem, da so vnosi v njih fiksni in se ne spreminjajo. Spremembe lahko ob morebitnem primeru dodajanja novih tipov meritev opravi le programer.

Question, QuestionType, Answer

Entiteta *Question* predstavlja posamezno vprašanje zdravniškega pregleda. Poleg vrstnega reda sta tu naslov in opis vprašanja. Tabela je povezana z entitetama *QuestionType* in *Answer*. V prvi najdemo podatke o tipu vprašanja, to je ali gre za vnosno polje, enojno izbiro itd. V drugi tabeli pa se v primeru tipov enojna in večkratna izbira hranijo možnosti oz. pripravljeni odgovori, med katerimi lahko pacient izbira. Tudi tabela *QuestionType* je temneje obarvana, saj je, tako kot to velja za tabeli, omenjeni v prejšnjem razdelku, ni mogoče spreminjati preko aplikacije.



Slika 4.2: Konceptualni model podatkovne baze

SequenceRule, QuestionRule, MeasurementRule

Vsakemu koraku je potrebno določiti, kateri izmed korakov mu sledi. To se naredi z vnosi v tabeli *SequenceRule*, kjer se definira tip pravil (ali morajo za nadaljevanje na naslednji korak med pripadajočimi pogoji veljati vsi ali le en), vrstni red pravil ter vrstni red naslednjega koraka. Zadnji podatek služi le v pomoč pri generiranju pregledov, saj sicer ta entiteta vsebuje dve povezavi na tabelo *Step*. Prva povezava definira pravila za trenuten korak, druga pa za korak, ki bo sledil.

Pogoji, ki povedo, katero izmed vejitvenih pravil se bo uporabilo, so definirani v *MeasurementRule* in *QuestionRule*. V teh tabelah se hranita tip pogoja (npr. je manjše, večje, enako) in vrednost, po kateri se bo opravila primerjava. Ker naša arhitektura dovoljuje več vprašanj in meritev v enem koraku, imata tabeli še povezavi na *MeasurementParameter* in *Question*, s katerima se da določiti, za katero meritev oziroma vprašanje na tem koraku gre.

Tabele za shranjevanje pacientovih vnosov

Tabele *ProcessInstance*, *MeasurementInstance*, *QuestionInstance*, *MParamInstance*, *QParamInstance* hranijo rezultate pacientovih meritev in odgovorov na vprašanja. Ker entitete ne hranijo podatkov o strukturi zdravniških pregledov in se tako razlikujejo od ostalih, so na načrtu podatkovnega modela obarvane z rumeno barvo.

Ko začne pacient z opravljanjem zdravniškega pregleda, se ustvari nova instanca v tabeli *ProcessInstance*. Da se ve, za katero verzijo pregleda gre, je ta povezana s tabelo *ProcessVersion* in hrani status, ki opisuje uspešnost opravljanja, ter datum in čas začetka in konca opravljanja pregleda. Vsaka procesna instanca ima lahko več instanc meritev in vprašanj, datum in čas vnosa/odgovora se hranita v *MeasurementInstance* in *QuestionInstance*. Da je mogoče ugotoviti, za rezultat katerega vprašanja ali meritve gre, sta tabeli povezani z entitetama *Measurement* in *Question*. V *MeasurementInstance* je še podatek o tem, ali je bila meritev vitalnega znaka ročno vnešena ali pa

je bila izmerjena z napravo, ki direktno komunicira z aplikacijo in vrednost samodejno doda k rezultatom. Dejanske vrednosti meritev in odgovorov na vprašanja so shranjene v tabelah *MParamInstance* in *AnswerInstance*.

4.3.2 Predstavitev zdravniškega pregleda v formatu JSON

Podatki, ki se izmenjujejo med zalednimi sistemi in odjemalskimi aplikacijami, so predstavljeni v formatu JSON. Za format JSON, namesto katerega drugega (npr. XML), smo se odločili predvsem zaradi njegove preprostosti pa tudi zaradi uporabe programskega jezika Typescript, ki omogoča enostavno in učinkovito upravljanje z objekti JSON.


```
{
   "id":182,
   "version":1,
   "active":false,
   "activated":false,
   "createdAt":"2017-07-21T11:45:44.821Z",
   "updatedAt":"2017-07-21T11:45:44.821Z",
   "repoId":null,
   "procDefId":131,
   "steps":[{
         "id":767,
         "type":"question",
         "order":1,
         "procVerId":182,
         "questions":[{
               "id":261,
               "description":"<p>Ali ste pred eno uro zauzili obrok?</p>",
               "title":"Zauzitje obroka",
               "order":1,
               "stepId":767,
               "questionTypeId":2,
               "answers":[{
                     "id":234,
                     "value":"da",
                     "questionId":261
                  },{
                     "id":235,
                     "value":"ne",
                     "questionId":261
                  }
```

```
31                  ],
32                  "questionType":{
33                     "id":2,
34                     "name":"SINGLE_CHOICE",
35                     "description":"Enojna izbira"
36                  }
37               }
38            ],
39            "seqRules":[{
40                  "id":583,
41                  "type":"AND",
42                  "order":0,
43                  "currStepId":767,
44                  "nextStep":{
45                     "id":768,
46                     "type":"measurement",
47                     "order":2,
48                     "procVerId":182
49                  },
50                  "questionRules":[{
51                        "id":103,
52                        "condition":"EQUALS",
53                        "value":"da",
54                        "seqRuleId":583,
55                        "questionId":261
56                     }
57                  ]
58               },{
59                  "id":584,
60                  "type":"AND",
61                  "order":1,
62                  "currStepId":767,
63                  "nextStep":{
64                     "id":769,
65                     "type":"measurement",
66                     "order":3,
67                     "procVerId":182
68                  }
69               }
70            ]
71         }
72      ]
73   }
```

Programska koda 4.1: Zdravniški pregled v obliki JSON

Izsek 4.1 prikazuje okleščeno predstavitev procesne verzije v obliki JSON.

Gre za pregled, ki vsebuje dva koraka. Prvi predstavlja vprašanje, ki pacienta sprašuje, ali je pred eno uro zaužil obrok. V kolikor je odgovor pozitiven, sledi prehod na drugi korak, kjer se opravi merjenje sladkorja v krvi. V nasprotnem primeru se pregled zaključi. Zaradi velike dolžine JSON predstavitve zdravniškega pregleda so na izseku le podatki o procesni verziji in o koraku z vprašanjem. Je pa iz prikazanih podatkov mogoče pridobiti dovolj dobro sliko o predstavitvi pregledov v tej obliki.

V prvih osmih vrsticah so podatki o procesni verziji. Vidi se, da je to prva verzija, ki ni bila še nikoli aktivirana ter da je bila objavljena le lokalno, saj vrednost *repoId* še ni definirana. Dalje sledi množica korakov (*steps*), ki so v tem primeru štirje: začetek, vprašanje, meritev in konec. Zaradi že omenjenih razlogov je tu prikazan le drugi korak. Parameter *type* pove, da gre za vprašanje, določen je tudi vrstni red. Korak je tu sestavljen iz enega vprašanja, ki se nahaja pod parametrom *questions*. Prikazani so naslov, opis in predefinirani možni odgovori. Opis je shranjen v označevalnem jeziku HTML, kar omogoča lepšo in bolj pregledno predstavitev na pacientovi napravi. Zatem pridejo na vrsto vejitvena pravila (*seqRules*). Pri tem koraku sta definirani dve pravili. Prvo se izvede, ko je vrednost pacientovega odgovora na vprašanje enaka „da“. Ta pogoj je viden pod parametrom *questionRules*. Lastnost *nextStep* nosi informacijo o tem, v kateri korak se bo pacient pomaknil s pritrdilnim odgovorom – v tem primeru bo to drugi korak po vrsti, sestavljen pa je iz meritve. Če ne bo prišlo do ujemanja pri prvem pravilu, se bo preverilo naslednje, ki pa ne vsebuje pogojev. Zgodil se bo premik na tretji korak, ki je tudi tipa *meritev*, vendar gre tu za zadnji, zaključni korak.

Pred začetkom implementacije rešitve se je pojavilo vprašanje, ali je bolje, da pacientova aplikacija pridobi zdravniški pregled v celoti ali pa da se pošilja vsak korak posebej in da za orkestracijo skrbi strežnik. Ker predpostavljamo, da velika večina pregledov ne bo izjemno kompleksna in ker bi se s tem lahko zelo povečala komunikacija med odjemalcem in strežnikom, kar bi bolj obremenjevalo strežnik, smo se odločili, da se pacientovi aplikaciji posre-

duje celoten pregled naenkrat na način, kot to izgleda na primeru 4.1. Zaradi strukture formata, s katerim je predstavljena procesna definicija, ki omogoča enostavno prikazovanje korakov in določanje, kako si ti sledijo, lahko za orekstracijo skrbi aplikacija za paciente sama.

4.4 Zaledni sistemi aplikacije

Za implementacijo zalednih sistemov je bilo uporabljeno okolje Node.js z dodatnimi ogrodji, kot sta naprimer Express in Sequelize. Koda je napisana v programskem jeziku Typescript.

Zaledni del je v osnovi razdeljen na štiri dele: del, ki implementira storitve REST, del s poslovno logiko (to so kontrolerji), del za komunikacijo s podatkovno bazo (servisi) in modeli, ki predstavljajo entitete v podatkovni bazi. Dalje se ti deli ločijo glede na namen – ali implementirajo logiko za pacientov ali zdravniški portal, komunikacijo z globalnim repozitorijem, pretvarjanje podatkov iz FHIR formata ...

4.4.1 Izpostavljanje virov REST

Glavni razred, ki izpostavlja vire REST, se imenuje *MainRoutes* in vsebuje tri različne funkcije. Prva izpostavlja dostopne točke za zdravniški portal, osnovni URI za te vire je `http://www.izbrana_domena.si/doctor`. Druga je namenjena portalu za paciente (`http://www.izbrana_domena.si/patient`). Tretja pa izpostavlja vire za komunikacijo z repozitorijem `http://www.izbrana_domena.si/repository`. Na omenjenih dostopnih točkah so implementirane akcije CRUD, preko katerih spletni vmesnik komunicira z zalednimi sistemi.

```
1 public static createDoctorRoutes(router: Router) {
2     const docBaseUrl = "/doctor";
3
4     router.get(docBaseUrl + "/processDefinition/:id",
5       (req: Request, res: Response) =>
6         DocProcessCtrl
7           .getProcessDefinition(req, res));
```

```
8
9       router.post(docBaseUrl + "/processDefinition",
10        (req: Request, res: Response) =>
11          DocProcessCtrl
12            .createProcessDefinition(req, res));
13
14      router.delete(docBaseUrl + "/processDefinition/:id",
15        (req: Request, res: Response) =>
16          DocProcessCtrl
17            .deleteProcessDefinition(req, res));
18  }
```

Programska koda 4.2: Del funkcije, ki izpostavlja vire REST.

Primer programske kode, ki izpostavlja vire za pridobivanje, ustvarjanje in brisanje zdravniških pregledov, je viden na izseku 4.2. Prikazana funkcija implementira krmilnike dogodkov, ki reagirajo na klice HTTP. V primeru zahtevka GET na naslov `http://www.izbrana_domena.si/doctor/processDefinition/1` se bo prožil krmilnik dogodkov v četrti vrstici. v objektu *req* bo shranjen id, ki je v tem primeru enak 1. Kot je razvidno v peti in šesti vrstici, se bo nato izvedel klic funkcije *getProcessDefinition* v razredu *DocProcessDefinitionCtrl*, ki bo vrnila odgovor s procesno definicijo z ustreznim id-jem. Krmilnik v štirinajsti vrstici, ki poskrbi za izbris procesne definicije, se proži ob zahtevku DELETE na taisti naslov, koda v deveti vrstici, ki je namenjena dodajanju nove procesne definicije, pa se izvede ob zahtevku POST na naslov `http://www.izbrana_domena.si/doctor/processDefinition`. Seveda funkcija *createDoctorRoutes* vsebuje še mnogo drugih krmilnikov, ki pa zaradi podobnosti z že predstavljenimi, niso prikazani.

4.4.2 Shranjevanje zdravniškega pregleda

Shranjevanje celotnega zdravniškega pregleda, torej tako procesne definicije kot tudi procesne verzije, poteka tako, da se na v prejšnjem razdelku predstavljen naslov, pošlje zahtevek POST, ki v telesu vsebuje zdravniški pregled, predstavljen v obliki JSON. Ko pride zahtevek POST na ustrezen naslov, se izvede krmilnik, prikazan v deveti vrstici na izseku kode 4.2, kjer se nato

kliče funkcija *createProcessDefinition* iz razreda *DocProcessCtrl*. Funkcija je prikazana na odseku 4.3.

```
1 public static createProcessDefinition(req: Request, res: Response) {
2       let processDefinition = req.body;
3       if(this.check(processDefinition)) {
4           DocProcessService
5               .saveProcessDefinition(processDefinition)
6               .then(
7                   procDef => res.status(200).send(procDef),
8                   err =>
9                     this.handleError("Creating process definition failed.",
10                    err)
11              );
12      }
13      else
14          return this.handleError("Creating process definition failed.",
15           "Error when parsing body from request.")
16 }
```

Programska koda 4.3: Funkcija, ki se izvede ob klicu za ustvarjanje procesne definicije.

Najprej se iz telesa zahtevka izlušči procesno definicijo, nato pa sledi klic funkcije, ki preveri morebitne napake v prejetih podatkih. V primeru, da so podatki ustrezni, se izvede klic funkcije *saveProcessDefinition*, ki se nahaja v razredu *DocProcessService*. V kolikor pa podatki ne ustrezajo pravilom, oziroma pride pri shranjevanju do napake, se kliče funkcija *handleError*, ki vrne ustrezno poročilo o napaki. Razred *DocProcessService* sicer sestavljajo funkcije, ki skrbijo za shranjevanje objektov v podatkovno bazo in pridobivanje le-teh iz nje. Funkcija *saveProcessDefinition* je predstavljena na izseku 4.4 in sprejme objekt, ki predstavlja procesno definicijo. Zatem se kliče model *ProcDefModel*, ki predstavlja objektno predstavitev entitete *ProcessDefinition*. Nad tem modelom se nato izvede funkcija *create*, ki se ji poda objekt, ki se ga dodaja, in parametre, ki naj se iz tega objekta shranijo. Objekt *processDefinition* lahko tako vsebuje še druge lastnosti, shranili se bodo pa le *title*, *description*, *patientDescription* in *chronicDiseaseId*. V kolikor je vstavljanje procesne definicije v podatkovno bazo uspešno, se kot rezultat vrne vstavljeni objekt *savedProcDef* skupaj z id-jem, ki se mu je določil. Iz prvotnega

objekta *processDefinition* se nato izlušči še procesna verzija, ki se ji nastavi pridobljeni id procesne definicije. Sledi klic funkcije *saveProcessVersion*, ki poskrbi, da se v podatkovno bazo vstavi še procesna verzija s pripadajočimi koraki, vprašanji, meritvami ... Funkcija je zaradi asinhronega shranjevanja, ko se programska koda izvaja naprej še pred zaključkom shranjevanja oz. zaključkom klica neke funkcije, zelo kompleksna, zato je tu ne bomo predstavili. Sestavljena je iz več klicev funkcij *create* nad modeli in uporabe Promisev, ki omogočajo izvajanje kode šele ob zaključku asihnronih funkcij.

```
1 public static saveProcessDefinition(processDefinition: ProcessDefinition) {
2       return ProcDefModel
3           .create(processDefinition,
4                   {fields: ['title', 'description',
5                   'patientDescription', 'chronicDiseaseId']})
6           .then(savedProcDef => {
7               let procVer = processDefinition
8                 .procVersions[processDefinition.procVersions.length - 1];
9
10                procVer.procDefId = savedProcDef.id;
11                return this.saveProcessVersion(procVer).then(ver => {
12                   savedProcDef.procVersions = [ver];
13                   return savedProcDef;
14              },
15              err => this.handleError("Inserting process version failed.",
16                     err)
17              );
18          },
19          err => this.handleError("Inserting process version failed.", err)
20          );
21 }
```

Programska koda 4.4: Funkcija za ustvarjanje procesne definicije

4.4.3 Shranjevanje vprašalnika FHIR

Standard FHIR je namenjen temu, da omogoča enovito izmenjavo informacij v zdravstvu, saj definira obliko podatkov, ki se izmenjujejo. Uporaba zagotavlja integracijo zdravstvenih storitev med različnimi ekipami in organizacijami. Izmenjava podatkov poteka preko storitev REST, lahko pa se uporabi tudi storitve SOAP, ki omogočajo boljšo varnost [8].

Rešitve FIHR so zgrajene iz množice modularnih komponent oz. t.i. virov (Resources). Te komponente si je mogoče predstavljati kot paprine forme, namenjene različnim kliničnim in administrativnim informacijam, ki se jih lahko zajema. Obstajajo različne vrste komponent, ki omogočajo zajem različnih informacij (npr. zajem podatkov o pacientu, zdravilih, alergijah ...).

Sprva smo želeli za izmenjavo zdravstvenih pregledov med zalednimi sistemi in pacientovo aplikacijo omogočiti tudi standard FHIR. Zdravstvene preglede, ki se jih ustvarja v naši rešitvi, bi lahko predstavili z uporabo komponente *Questionnaire* [26], ki jo sestavlja organizirana množica vprašanj, katerih namen je pridobiti določene informacije od pacientov (primeri uporabe so npr.: vprašalnik o družinskih boleznih, vprašalnik o zdravstveni zgodovini pacienta ali pa razni raziskovalni vprašalniki). Pacientove rezultate pa bi lahko predstavili s *QuestionnaireResponse* [28].

Po pregledu omenjenih možnosti pa smo naleteli na težave, ki so nas privedle k temu, da smo za izmenjavo zdravniških pregledov uporabili le format, prikazan v razdelku 4.3.2. Problemi se pojavijo pri vejitvah, saj je pri našem tipu pregledov mogoče iz enega koraka nadaljevati v več drugih. Vsak korak ima tako pravila, ki povedo, na kateri korak se bo premaknil pacient, glede na vrednost meritve ali odgovora. Za eno vejitev (torej pravilo iz enega koraka v drugega) je mogoče določiti več pogojev, ki morajo biti izpolnjeni, da se bo prehod zgodil. Npr. določi se lahko dva pogoja, da mora biti srčni utrip višji od 60 in nižji od 90 udarcev na minuto. Definira pa se lahko tudi, ali med pogoji velja konjkuktivna ali disjuktivna odvisnost. Pri uporabi vira FHIR, ki se imenuje *Questionnaire*, pa posameznemu vprašanju ni mogoče dodati podatkov o tem, kateri korak je naslednji, ampak je to narejeno z obratne strani. Vsakemu vprašanju se lahko določi, kdaj je „omogočeno“ (z uporabo parametra *enableWhen*), tj. ali se pacientu prikaže ali ne. Težava je tako v tem, da je treba pri tem načinu predstavitve pri vsakem koraku pregledovati naslednje korake in pripadajoče lastnosti ter tako ugotoviti, kateri korak sledi, oz. je omogočen, in se ga lahko prikaže pacientu. To lahko rezultira v tem, da je potrebno pregledati več korakov oziroma da se mora pri ustvarjanju

vprašalnika natančno določiti vrstni red korakov, da se bo lahko „omogočen“ korak našlo čimprej. To pa ni vedno enostavno. Pri trenutni predstavitvi to ni potrebno, ampak se takoj ve, kaj sledi. Problem je tudi, da pri formatu FHIR ni mogoče določiti, ali med pogoji velja konjuktivna ali disjuktivna odvisnost. Morda bi se to sicer lahko implementiralo z uporabo t. i. razširitev [7], vendar raziskovanje teh možnosti že presega okvire našega diplomskega dela.

Zato smo namesto tega, da bi standard FHIR uporabili za predstavitev zdravniških pregledov, aplikaciji dodali storitev REST, ki omogoča pretvorbo vprašalnikov FHIR, to je vprašalnikov, predstavljenih z virom *Questionnaire*, v naš tip zdravstvenih pregledov in njihovo shranjevanje v podatkovno bazo. Ker so ti vprašalniki precej splošno definirani, smo pretvarjanje implementirali le za nekaj tipov. Pomagali smo si s primeri vprašalnikov, ki jih je mogoče najti na spletni strani [27]. Zdravniški pregled, ki smo ga predstavili v razdelku 4.3.2, lahko z uporabo vira *Questionnaire* izgleda tako kot na izseku 4.5.

```
1 {
2   "resourceType": "Questionnaire",
3   "id": "3141",
4   "version":1,
5   "title": "FHIR merjenje sladkorja v krvi eno uro po kosilu.",
6   "description": "Pregled je namenjen merjenju sladkorja v krvi. Meritev
7                   mora biti opravljena eno uro po zauzitju vecjega obroka.",
8   "url": "http://hl7.org/fhir/Questionnaire/3141",
9   "status": "active",
10  "date": "2017-07-07",
11  "item": [
12    {
13      "linkId": "0",
14      "text": "Ali ste pred eno uro zauzili obrok?",
15      "type": "choice",
16      "required": true,
17      "option":[
18        {
19          "valueCoding": {
20            "code": "Da"
21          }
22        },
23        {
```

```
24            "valueCoding": {
25              "code": "Ne"
26            }
27          }
28        ]
29      },
30      {
31        "linkId": "1",
32        "text": "Meritev sladkorja v krvi",
33        "type": "decimal",
34        "required": true,
35        "code": [
36          {
37            "system": "http://loinc.org",
38            "code": "2339-0"
39          }
40        ],
41        "enableWhen": [{
42          "question": "0",
43          "answerString": "Da"
44        }]
45      }
46    ]
47  }
```

Programska koda 4.5: Primer vprašalnika, predstavljenega z virom *Questionnaire*

Parameter *resourceType* pove, za kateri tip vira gre (v tem primeru *Questionnaire*). Prikazani so tudi naslov, opis, verzija in datum zadnje spremembe. *Url* predstavlja globalno unikaten URI, določen temu vprašalniku. Polje *status* ima lahko štiri različne vrednosti: *draft*, *active*, *retired*, *unknown*. Prva pomeni, da se vprašalnik še vedno spreminja (ustvarjanje še ni zaključeno), druga, da je vprašalnik pripravljen za uporabo, tretja pa pomeni, da se ga ne sme več uporabljati. Zadnja vrednost kaže, da sistem zaradi neznanega razloga ne ve natačno, kateri status temu vprašalniku pripada.

Vprašalnik je sestavljen iz več elementov *item*, ki predstavljajo vprašanja oziroma skupine vprašanj, saj se te elemente lahko gnezdi. Parameter *linkId* predstavlja unikaten id, dodeljen nekemu elementu *item*. Obstaja več tipov vprašanj: *group* – predstavlja skupino vprašanj, *display* – element, namenjen le prikazu, *choice* – vprašanje, kjer mora pacient kot odgovor izbrati eno

izmed ponujenih možnosti, *decimal* – decimalno število, *integer* – celo število, *date* – datum ... Tip se določi s poljem *type*. Vsak *item* vsebuje tudi opis v polju *text* in podatek o tem, če je korak obvezen ali ne (*required*).

S poljem *code*, prikazanem pri drugem koraku, ki predstavlja meritev, se lahko definira, za kateri tip meritve oziroma vprašanja gre, pri čemer se za definicijo uporablja kode, določene z mednarodnim standardom *LOINC (Logical Observation Identifiers Names and Codes)*. To je standard, ki rešuje problem različne identifikacije istih testov in meritev med različnimi ustanovami. Namesto tega je za vsak tip meritve določena unikatna koda, ki omogoča prepozavanje podatkov o meritvi med velikim številom različnih sistemov [15]. Za prikazani primer koda „2339-0“ predstavlja meritev sladkorja v krvi.

Kot je že omenjeno, parameter *enableWhen* omogoča definicijo vejitev. V tem primeru mora za vprašanje (*question*) z id-jem enakim nič veljati, da bo pacientov vnos enak „Da“. Če bo ujemanje pozitivno, bo ta korak prikazan, sicer pa ne.

S parametrom *option* se lahko definira odgovore, med katerimi mora pacient izbrati (uporabno pri vprašanju tipa *choice*). Primer uporabe tega parametra se vidi pri prvem vprašanju, od petnajste vrstice naprej, kjer sta določena možna odgovora „Da“ in „Ne“.

S kodami *LOINC* je poleg meritev mogoče definirati tudi različne možne odgovore na vprašanja. Tako se lahko pri pregledih določi korak tudi na način, ki je viden na delu vprašalnika 4.6.

```
1 {
2       "linkId": "FeelingBadAboutSelf",
3       "code": [
4         {
5           "system": "http://loinc.org",
6           "code": "44258-2"
7         }
8       ],
9       "text": "Trouble concentrating on things, such as reading
10       the newspaper or watching television",
11      "type": "choice",
12      "required": true,
13      "options": {
```

```
14          "reference": "http://loinc.org/vs/LL358-3",
15          "display": "Patient Health Questionnaire (PHQ-9) Not at all/
16          Several days/More than half the days/Nearly every day"
17        }
18 }
```

Programska koda 4.6: Del vprašanja, definiranega z uporabo kode *LOINC*

Koda „44258-2“, ki se nahaja pod parametrom *code*, označuje možnosti odgovorov „Not at all“ – nikoli, „Several days“ – več dni, „More than half the days“ – več kot polovico dni, „Nearly every day“ – skoraj vsak dan, ki se jih lahko uporablja pri različnih vprašanjih. V predstavljenem primeru gre za možne odgovore na vprašanje o tem, ali ima pacient težave s koncentracijo, npr. z branjem časopisov ali gledanjem televizije. Ta način uporabe elementa *code* je sicer zelo uporaben, saj lahko precej skrajša dolžino vprašalnika in naredi strukturo bolj standardno. Tako lahko ta element prepoznajo vsi sistemi, ki implementirajo uporabo standarda *LOINC*, kar rešuje problem izolacije podatkov z lokalnimi identifikatorji, ki jih lahko pravilno prepozna in uporablja le sistem, ki jih je ustvaril. Ker bi morali, če bi hoteli prepoznavati takšne elemente, kot je prikazani, pri našem razčlenjevalniku vprašalnikov FHIR dodati še možnost mapiranja različnih vprašanj s kodami *LOINC* in ustrezno prepoznavanje možnih odgovorov, kar lahko postane zelo kompleksno, smo implementirali le razčlenjevanje tistih tipov vprašanj *choice*, ki možne odgovore podajo na način kot pri prvem vprašalniku 4.5. Je pa dodano prepoznavanje kod, ki predstavljajo tipe meritev, ki se jih lahko uporablja pri ustvarjanju našega tipa vprašalnikov.

Sicer pa naša storitev REST za pretvarjanje vprašalnikov FHIR najprej pregleda osnovne podatke o vprašalniku (naslov, opis, status, verzija), dalje pa sledi razčlenjevanje delov vprašalnika (*items*). Razčlenjevalnik prepozna različne tipe vprašanj (osnovne tipe, kot so vprašanja, ki pričakujejo kot odgovor celo ali decimalno število, datum, čas, niz znakov; pa tudi kompleksnejše tipe, kot je izbira z enim možnim odgovorom) in jih ustrezno pretvori v našo obliko. Meritve se prepozna po kodah *LOINC*. Storitev tako prepozna več tipov vprašalnikov in jih pretvori v format, prikazan v razdelku 4.3.2.

Potem se tako predstavljen pregled shrani v podatkovno bazo.

4.4.4 Globalni repozitorij zdravniških pregledov

Aplikaciji za ustvarjanje pregledov sta dodani funkcionalnosti, ki nudita možnosti prenosa zdravniških pregledov iz repozitorija in nalaganje pregledov vanj. Nek zdravnik se lahko odloči, da bo pregled, ki ga je ustvaril, naložil v repozitorij in tako omogočil drugim, da si ga prenesejo in dodeljujejo svojim pacientom.

Globalni repozitorij bi bilo najbolje implementirati tako, da bi se sprogramiralo storitve, ki tečejo v oblaku in omogočajo sprejemanje in shranjevanje poslanih pregledov v globalno podatkovno bazo. V primeru prenašanja pregleda iz repozitorija se izvede klic na storitev, ki vrne pregled v formatu JSON. Ta se nato shrani v lokalno podatkovno bazo neke ustanove. Pri naši implementaciji smo namesto dodajanja storitev v oblak, omenjeno arhitekturo posnemali tako, da smo lokalno postavili še eno verzijo podatkovne sheme s skoraj v celoti enakimi tabelami, kot so prikazane na sliki 4.2. Razlike so le v tem, da tabeli *ProcessDefinition* in *ProcessVersion* ne vsebujeta stolpcev *repoId* in *repoStatus*, ampak ima *ProcessVersion* še dodatna stolpca *avgRating* in *downloads*, v katerih se za vsako verzijo hranita povprečna ocena uporabnikov in število prenosov. Dodana je še tabela *Review*, ki hrani komentarje in ocene zdravniških pregledov v repozitoriju.

Projektu smo dodali storitve, ki upravljajo s to podatkovno shemo in omogočajo shranjevanje zdravniških pregledov vanjo. To poteka tako, da se na vir REST, ki je dostopen na naslovu `http://www.izbrana_domena.si/doctor/repository/new`, pošlje zahtevek POST z informacijama o id-jih procesne definicije in verzije, ki se nalagata v repozitorij. Glede na ti dve informaciji se iz podatkovne baze najprej pridobi ustrezna procesna definicija, nato pa še verzija. Te se doda v telo zahtevka POST, ki se ga pošlje na naslov `http://www.izbrana_domena_repozitorij.si/repository/new`, na katerem je dostopna storitev REST, ki skrbi za vstavljanje prejetega pregleda v podatkovno bazo globalno dostopnih procesnih definicij. Po uspešnem

shranjevanju se vrne odgovor, ki vsebuje *repoId*-ja vstavljene definicije in verzije. Zgodita se posodobitvi lokalne definicije in verzije zdravniškega pregleda. Procesni definiciji se vrednost stolpca *repoId* nastavi na pridobljeno vrednost, *repoStatus* pa postane „uploaded“, kar pomeni, da je bil ta pregled že naložen v repozitorij. Prav tako stolpec *repoId* v tabeli *ProcessVersion* dobi vrednost id-ja globalne procesne verzije. Lokalna procesna definicija je tako lahko treh tipov – če ne vsebuje nobenih podatkov o repozitoriju, pomeni, da je objavljena le lokalno; če je shranjen *repoId* in je *repoStatus* enak „uploaded“, je bila vsaj ena izmed verzij pregleda naložena v repozitorij (to omogoča ustvarjalcu, da javno objavlja še nove verzije, ki jih ustvari lokalno); v primeru, da obstaja *repoId* in je *repoStatus* „downloaded“ pa pomeni, da je bil pregled prenešen iz repozitorija (zdravnik lahko dodaja nove lokalne verzije, ne more jih pa naložiti v repozitorij).

Prenos zdravniškega pregleda iz repozitorija v lokalno podatkovno bazo pa se začne z zahtevkom GET na naslov `http://www.izbrana_domena.si/repository/download?procDefId=x&procVerId=y`. Zahtevek se posreduje na naslov z drugačno domeno (`http://www.izbrana_domena_repozitorij.si/repository/download?procDefId=x&procVerId=y`), kjer se nahaja storitev REST za pridobitev zdravniškega pregleda iz globalne podatkovne baze. Glede na id-ja procesne definicije (x) in verzije (y) se vrne pregled v formatu JSON in poveča število prenosov. Vrnjen pregled se vstavi v lokalno podatkovno bazo, pri čemer se doda tudi podatke o id-jih iz repozitorija; *repoStatus* pa se nastavi na „downloaded“.

Dodane so tudi storitve, ki posameznim procesnim verzijam omogočajo dodajanje komentarjev in njihovo ocenjevanje. Tako lahko zdravniki izrazijo mnenje o pregledu, ki so ga prenesli iz repozitorija, dodajo morebitne kritike oz. predloge za izboljšavo in podajo neko oceno o tem, kako so z njegovo uporabo zadovoljni. To omogoča ostalim uporabnikom, da si o zdravniških pregledih pridobijo celovito sliko, spoznajo njihove prednosti in pomanjkljivosti ter ugotovijo, kateri izmed pregledov za spremljanje določene bolezni bi najbolj ustrezal njihovim potrebam.

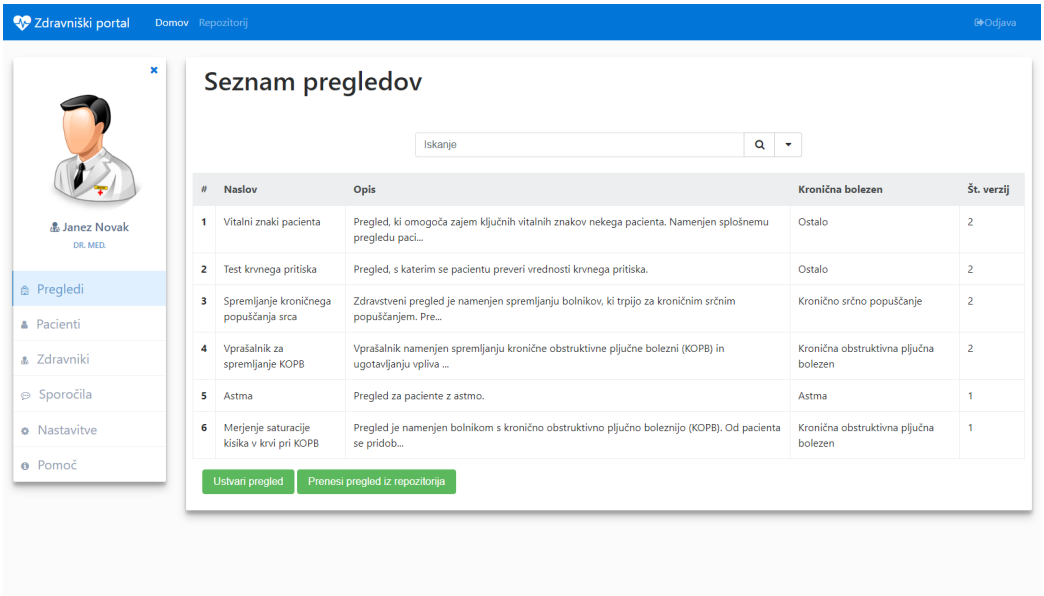
4.5 Spletna aplikacija (odjemalski del)

Obe spletni aplikaciji (tako zdravniški portal kot tudi aplikacijo za opravljanje pregledov) smo sprogramirali z uporabo platforme *Angular 4*. Za oblikovanje uporabniškega vmesnika smo uporabili ogrodje *Bootstrap 4*. Aplikacija tako teče v brskalniku in z uporabo klicev REST od zalednih sistemov pridobiva podatke za prikaz.

Struktura projektov obeh aplikacij je skoraj enaka z nekaj manjšimi razlikami. Aplikacijo sestavlja glavni modul (*module*), ki združuje več komponent (*components*). Vsaka komponenta predstavlja svoj pogled (*view*), ki se uporabniku prikaže na zaslonu. Npr. del aplikacije, na katerem se prikaže seznam zdravniških pregledov, je ena komponenta, pogled, kjer se ustvarja nov pregled, pa druga. Modeli (*models*) so razredi, ki imajo podobno vlogo kot modeli zalednih sistemov – predstavljajo entitete v podatkovni bazi oz. se lahko z njimi predstavijo objekti, pridobljeni od zalednih sistemov. Glaven namen modelov je ta, da se z njihovo uporabo lahko objektom določa tipe, kar omogoča bolj učinkovit in hitrejši razvoj aplikacije, saj je prepoznavanje napak precej lažje.

Servisi (*services*) so razredi, ki implementirajo logiko za komunikacijo z zalednimi sistemi, lahko pa se jih uporablja tudi npr. za prenos podatkov med različnimi komponentami aplikacije. Servis *Router* skrbi za navigacijo. Z njim se definira, katera komponenta bo aktivna glede na naslov, zahtevan v brskalniku. Ob zahtevku na naslov `http://www.izbrana_domena/processList` bo npr. aktivna komponenta *ProcessListComponent*, ki skrbi za prikaz seznama zdravniških pregledov.

4.5.1 Aplikacija za ustvarjanje pregledov



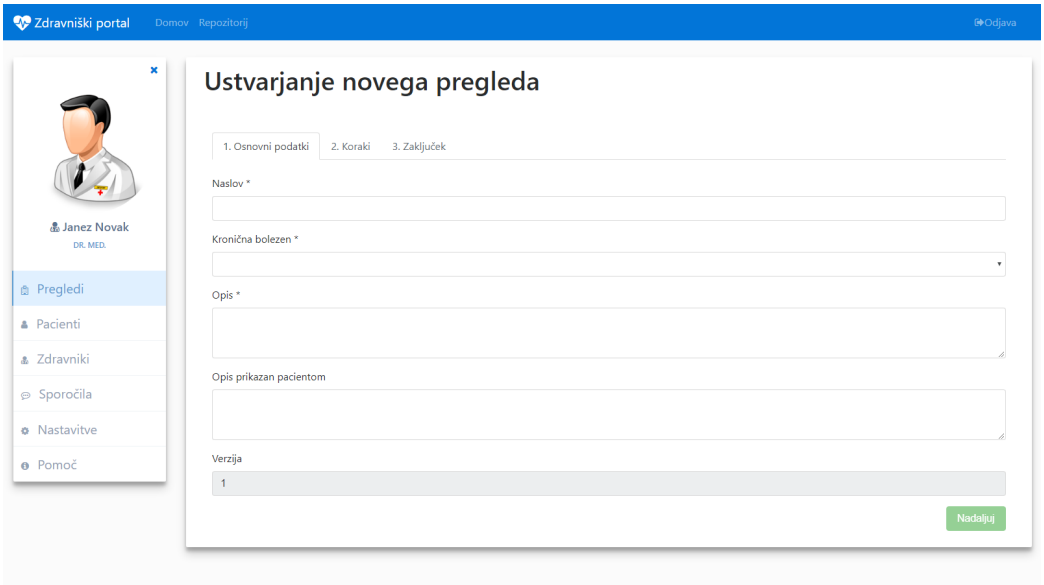
Slika 4.3: Glavna stran s seznamom zdravniških pregledov

Na sliki 4.3 je prikazano začetno okno aplikacije, ki se uporabniku – zdravniškemu delavcu pokaže po prijavi. Ker smo se pri programiranju osredotočali na druge funkcionalnosti, prijava v pravem pomenu besede še ni implementirana, saj za upravljanje z uporabniki in njihovimi gesli še ni poskrbljeno.

Vrhnji navigacijski menu (obarvan modro) ima le 3 možnosti: *Domov* in *Repozitorij*, skrajno desno pa je še gumb za odjavo uporabnika. Ob kliku na *Domov* se uporabnik nahaja na „lokalnem“ delu aplikacije s pregledi, ki se tičejo le oddelka oz. ustanove. Klik na *Repozitorij* pa prikaže okno z globalno dostopnimi zdravniškimi pregledi, ki jih lahko uporabnik prenese v lokalno podatkovno bazo. Ta del je podrobneje predstavljen v poglavju 4.5.2. Na levi strani pogleda se nahaja navigacijski menu, na katerem lahko uporabnik izbira med različnimi podstranmi aplikacije. Glede na to, na kateri podstrani se uporabnik nahaja, se del menuja, s povezavo na ta del aplikacije, obarva modro. Trenutno se torej uporabnik nahaja na strani *Pregledi*. Nad poveza-

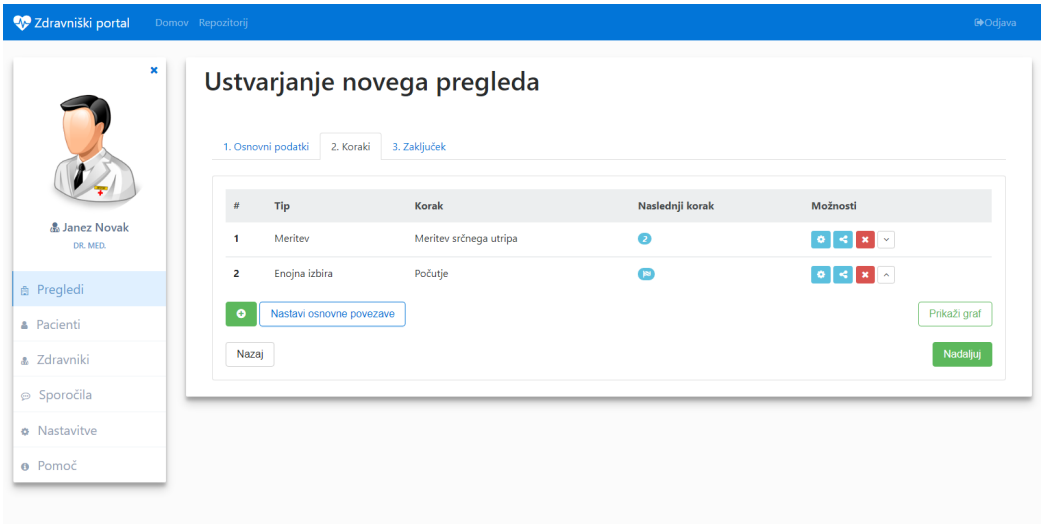
vami na druge dele aplikacije se izpišejo ime in priimek zdravnika ter njegov naziv; prikazana je tudi morebitna profilna slika. Navigacijski menu je viden tudi ob premiku na katero izmed podstrani, skrije se ga lahko s klikom na križec v zgornjem desnem kotu. Desno od levega navigacijskega menuja je postavljeno okno, kjer se prikazujejo izbrane podstrani.

Na pogledu *Seznam pregledov* je prikazana tabela zdravniških pregledov nekega oddelka ali ustanove. Tabelo sestavlja pet stolpcev: vrstni red; naslov; opis; tip kronične bolezni, za katero je pregled namenjen; število verzij pregleda. S klikom na vrstico tabele se zgodi preusmeritev na zaslonsko masko, vidno na sliki 4.10. Nad tabelo je vnosno polje, s katerim je mogoče iskati po naboru pregledov. S klikom na skrajno desni gumb (puščica navzdol) se prikažeta še dva spustna menuja, ki omogočata filtriranje pregledov po kronični bolezni in njihovo razvrščanje. Pod tabelo sta dva gumba; ob kliku na *Prenesi pregled z repozitorija* se odpre del aplikacije, ki prikazuje preglede, shranjene v repozitoriju, klik na *Ustvari pregled* pa prikaže okno, vidno na 4.4.



Slika 4.4: Ustvarjanje zdravniškega pregleda – podajanje osnovnih informacij

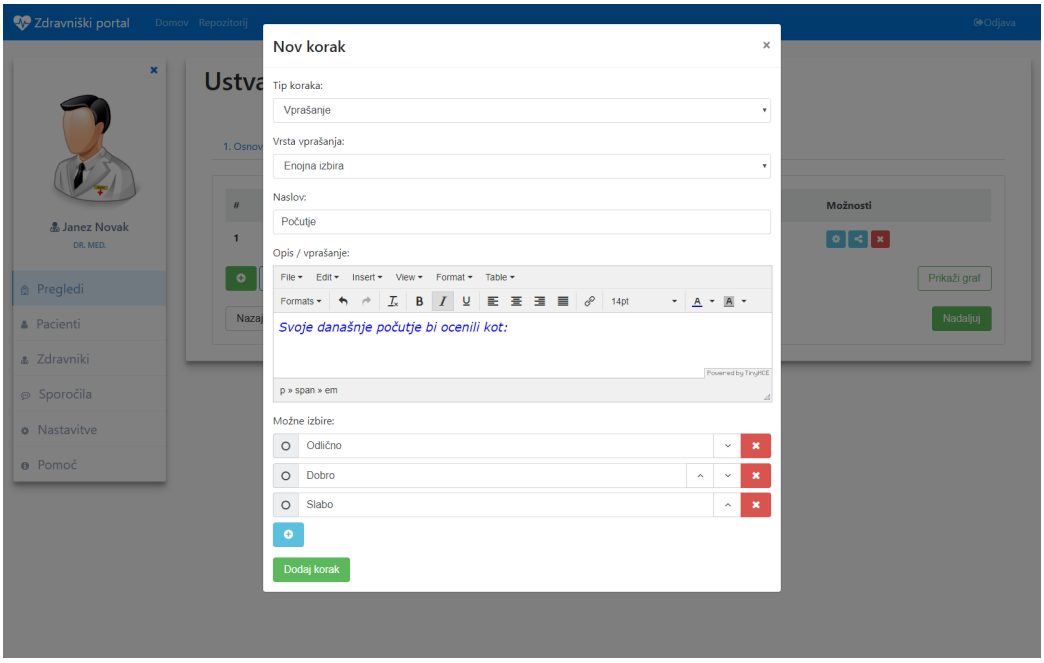
Ustvarjanje novega zdravniškega pregleda poteka v treh korakih. Na sliki je prikazan prvi korak, kjer je potrebno vpisati in izbrati osnovne podatke o zdravniškem pregledu. Obvezni podatki so označeni z zvezdico (*). Ko uporabnik izpolni vsa obvezna vnosna polja, postane gumb *Nadaljuj* omogočen, s čimer se lahko napreduje na naslednji korak na sliki 4.5, kjer se zdravniškemu koraku določi, iz katerih meritev in vprašanj je sestavljen.



Slika 4.5: Ustvarjanje zdravniškega pregleda – dodajanje korakov

Na zaslonski maski, ki predstavlja drugi del ustvarjanja pregleda, se nahaja tabela korakov, iz katerih je sestavljen nek zdravniški pregled. Tabela ima 5 stolpcev, ki predstavljajo vrstni red koraka, tip (*meritev*, *vprašanje* ali *množica*), ime koraka in seznam možnih naslednjih korakov (vejitve). Zadnji stolpec vsebuje gumbe, ki omogočajo urejanje posameznega koraka. Klik na prvi gumb odpre okno, kjer se lahko ureja osnovne lastnosti koraka (tip, naslov, opis ...), klik na drugega pa prikaže okno za dodajanje vejitev iz koraka. Tretji gumb je namenjen odstranjevanju korakov, zadnja dva gumba pa omogočata spreminjanje njihovega vrstnega reda. Pod tabelo se na levi strani nahaja gumb z znakom plus (+), ki omogoča dodajanje novih korakov. Zraven je gumb *Nastavi osnovne povezave*, s katerim se lahko korakom nastavi

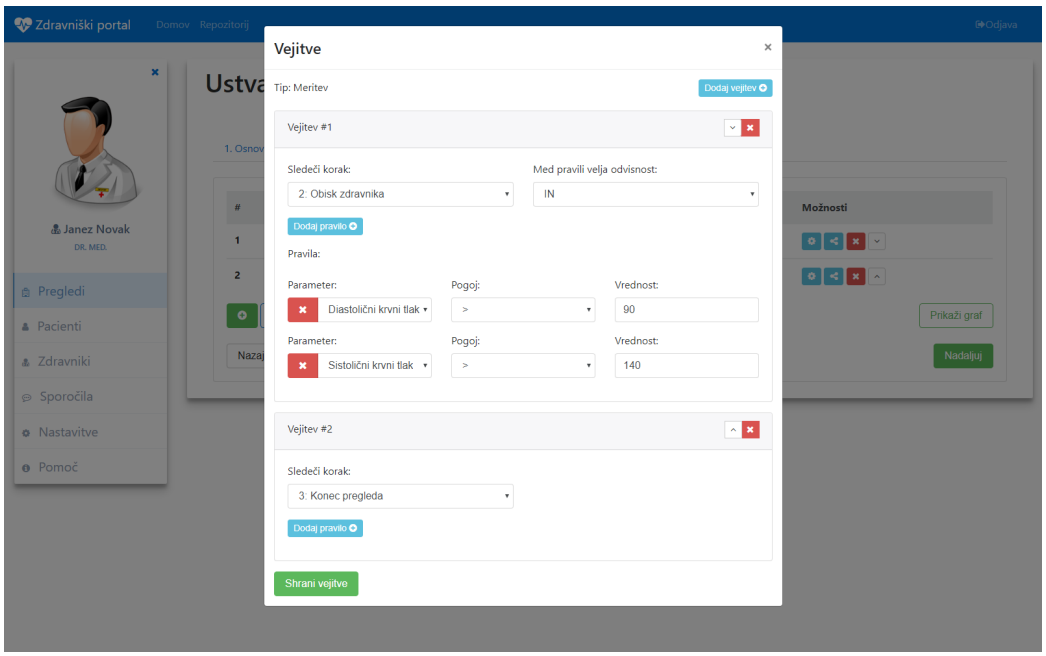
povezave tako, da si koraki sledijo po vrsti, eden za drugim. Skrajno desno v tej vrstici se nahaja še gumb *Prikaži graf*, ki omogoča predstavitev korakov in povezav med njimi na grafu. Primer grafičnega prikaza se lahko vidi na sliki 4.8. S tem si lahko uporabnik precej pomaga pri dodajanju vejitev, saj lahko hitro vidi, kako si koraki sledijo in katere povezave je morda še treba dodati, da pregled ne bo vseboval nedefiniranih stanj (stanje, ko iz nekega koraka ne vodi nobena pot oz. v nek korak ni mogoče priti). Na dnu sta še dva navigacijska gumba za premikanje med koraki ustvarjanja pregleda. *Nadaljuj* postane omogočen šele, ko uporabnik pregledu določi vsaj en korak.



Slika 4.6: Ustvarjanje zdravniškega pregleda – dodajanje koraka

Slika 4.6 prikazuje okno, ki se odpre ob kliku na gumb za dodajanje koraka. Na vrhu se najprej izbere tip koraka, ki je lahko *meritev* ali *vprašanje*. Čeprav sama arhitekura podpira tip *množica*, nam njegove uporabe zaradi kompleksnosti in posvečanja drugim ciljem še ni uspelo implementirati. Meritvi po izbiri tipa ni potrebno določati nobenih drugih lastnosti, vprašanju pa je treba najprej definirati vrsto vprašanja, nato pa določiti še naslov in

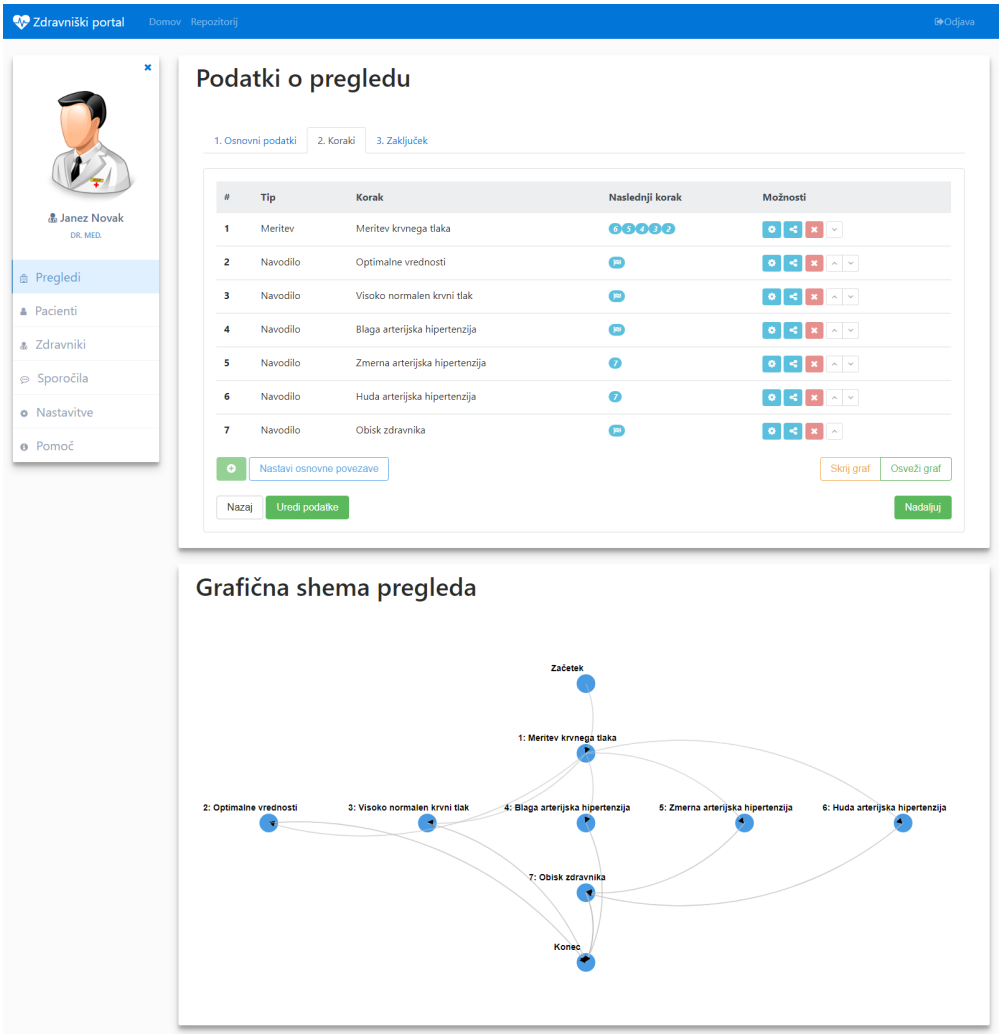
opis. Tega se lahko ureja z uporabo TinyMCE WYSIWYG urejevalnika [31]. Uporabnik lahko opis oz. vprašanje vizualno ureja in mu spreminja velikost, stil, postavitev, barvo črk ter dodaja slike, s tem pa omogoča pacientu lažje prepoznavanje vprašanja ali navodil. Če sta pri vrsti vprašanja izbrani *Enojna izbira* ali *Večkratna izbira*, se pod opisom pojavijo še vnosna polja, s katerimi se dodaja možne odgovore, med katerimi bodo lahko pacienti izbirali. S klikom na *Shrani* se korak doda zdravniškemu pregledu, okno pa se zapre.



Slika 4.7: Ustvarjanje zdravniškega pregleda – dodajanje vejitev

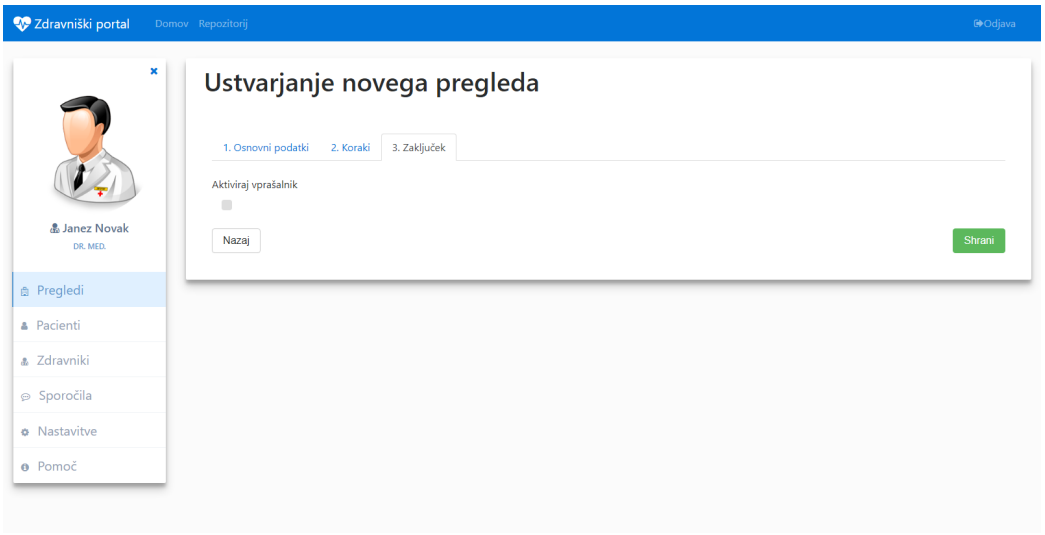
Na zaslonski maski 4.7 je prikazano okno za dodajanje vejitev. Možno je dodati več vejitev; z vsako se definira en naslednji korak, na katerega se bo ob izpolnjenih pogojih zgodil prehod. V kolikor pa želimo določiti pravila oz. pogoje, ki morajo veljati, da se bo prehod na ta korak zgodil, je treba opraviti klik na *Dodaj pravilo +*. Takrat se na desni pojavi vnosno polje, kjer se lahko izbere tip odvisnosti med pogoji (konjuktivna – IN, disjuktivna – ALI). Spodaj pa se prikažejo tri polja. S prvim se izbere parameter, glede na

katerega se bo izvajala primerjava. Drugo določa tip pogoja (večje, manjše, je enako), zadnji pa vrednost, s katero se bo pacientov vnos primerjal. Na sliki se bo tako prva vejitev upoštevala v primeru, ko bo diastolični krvni tlak višji od 90 mmHg in sistolični tlak višji od 140 mmHg. Tedaj bo pacient preusmerjen na drugi korak z naslovom *Obisk zdravnika*. V kolikor bosta (ena ali obe) vrednosti krvnih tlakov drugačni, pa se bo pregled zaključil. Vejitve se sicer pregledujejo po vrsti, torej najprej prva, nato druga itd. Vrstni red se spreminja z gumboma s puščicama v zgornjem desnem kotu vsake vejitve.



Slika 4.8: Primer zdravniškega pregleda za testiranje krvnega pritiska

Na zaslonskem posnetku 4.8 je mogoče videti primer tabele s koraki zdravniškega pregleda, namenjenega testiranju krvnega pritiska. Prikazana je tudi grafična predstavitev korakov in povezav med njimi. Krogci predstavljajo korake, puščice pa označujejo poti med njimi. Ker je zdravniški pregled že shranjen, je spreminjanje korakov onemogočeno. Zato so gumbi v stolpcu *Možnosti* osiveli, klik nanje ni mogoč. Prikazan pa je gumb *Uredi podatke*, ki omogoča urejanje korakov v primeru, da pregled še ni bil aktiviran. S klikom nanj postanejo namreč gumbi za urejanje in dodajanje vejitev ponovno omogočeni.

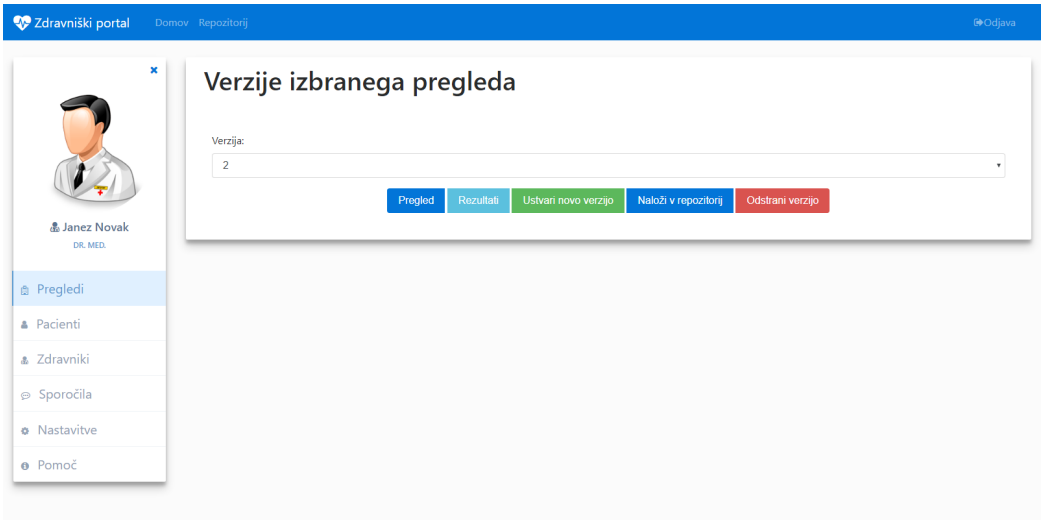


Slika 4.9: Ustvarjanje zdravniškega pregleda – zadnji korak

Zadnji, tretji korak ustvarjanja pregleda je prikazan na sliki 4.9. Tu se pregledu le še določi, ali je aktiviran in se ga shrani s klikom na *Shrani*.

V primeru, da na seznamu pregledov kliknemo na katero izmed vrstic v tabeli, se prikaže okno na sliki 4.10. Iz spustnega menuja se izbere verzijo zdravniškega pregleda, nad katero se potem lahko izvaja akcije, ki jih nudijo gumbi pod menujem. *Pregled* odpre strani za ustvarjanje pregleda, le da so vsa vnosna polja in tabela korakov izpolnjeni s podatki izbrane procesne verzije. Vsa polja so osivela oz. onemogočena in se jih ne more urejati.

Spreminjati je možno le osnovne podatke procesne definicije, korake pa se lahko ureja le neaktivni procesni verziji. Kakršnekoli spremembe se omogoči z gumbom *Uredi podatke*.

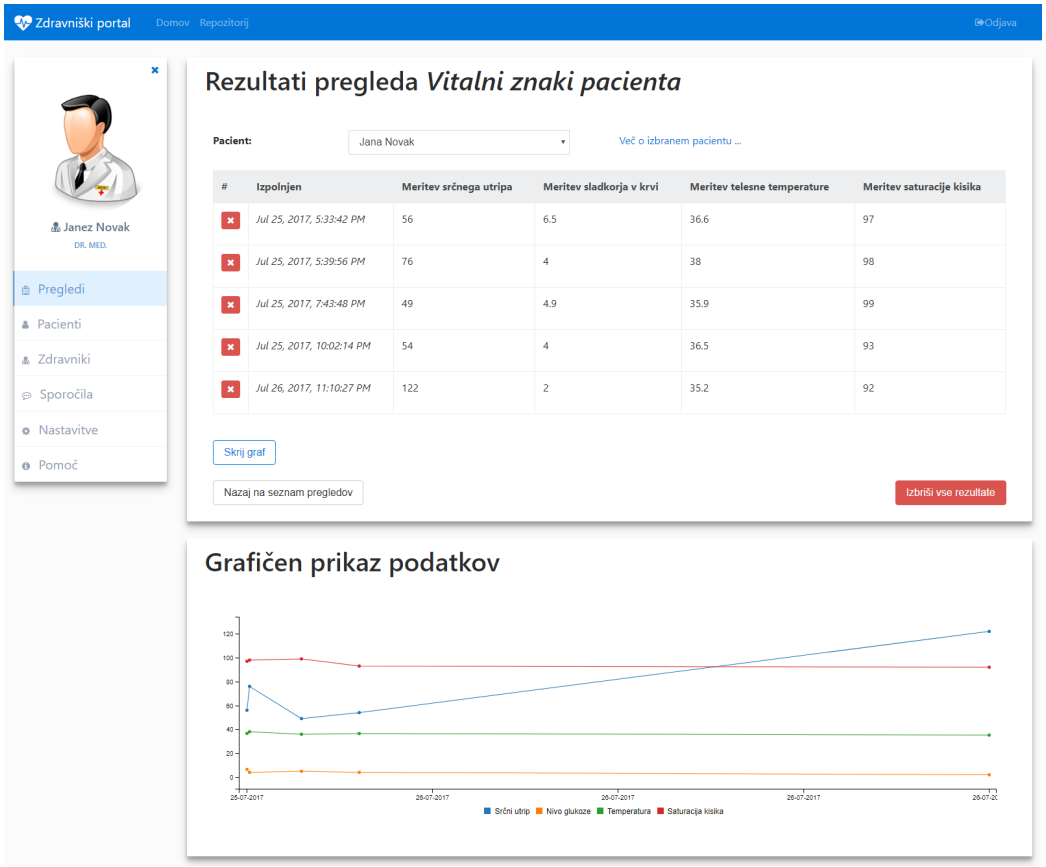


Slika 4.10: Pogled z izbiro verzije zdravniškega pregleda

Rezultati odprejo okno, kjer se prikazujejo opravljenje meritve in odgovori na vprašanja izbranega pacienta. Primer pogleda za pregled *Vitalni znaki pacienta*, ki vsebuje štiri meritve in nobenega vprašanja, je prikazan na sliki 4.11. Rezultati se prikažejo v tabeli, kjer vsaka vrstica predstavlja eno instanco (en izpolnjen pregled). Tabela sestoji iz toliko stolpcev, kot je meritev in vprašanj pregleda. Na začetku sta še dva dodatna stolpca – v prvem so gumbi za brisanje instanc, v drugem pa sta izpisana datum in čas oddaje opravljenega pregleda. V kolikor pregled vsebuje meritve, se ob kliku na gumb *Prikaži meritve na grafu* odpre okno z grafom, na katerem so prikazane vrednosti meritev skozi čas.

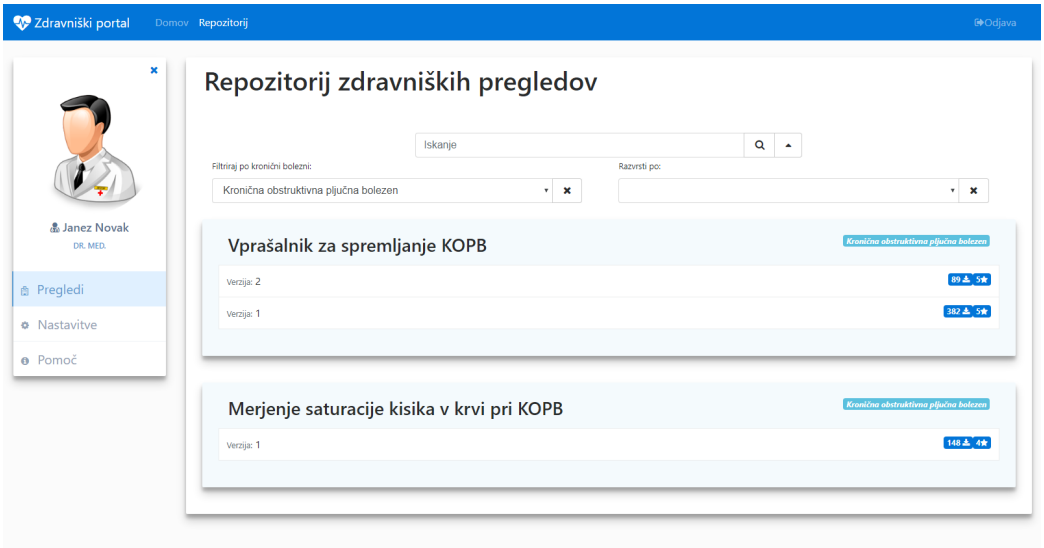
Na strani *Verzije izbranega pregleda* se lahko še ustvari novo verzijo zdravniškega pregleda, kar pomeni, da se zgodi preusmeritev na stran za ustvarjanje pregleda, pri čemer se polja predizpolnijo s podatki izbrane verzije, vsa vnosna polja pa so omogočena. Tako se lahko izbrano procesno verzijo

poljubno spremeni oz. nadgradi. Verzijo je mogoče tudi naložiti v globalni repozitorij ali pa jo izbrisati.



Slika 4.11: Rezultati opravljanja zdravniškega pregleda za določenega pacienta

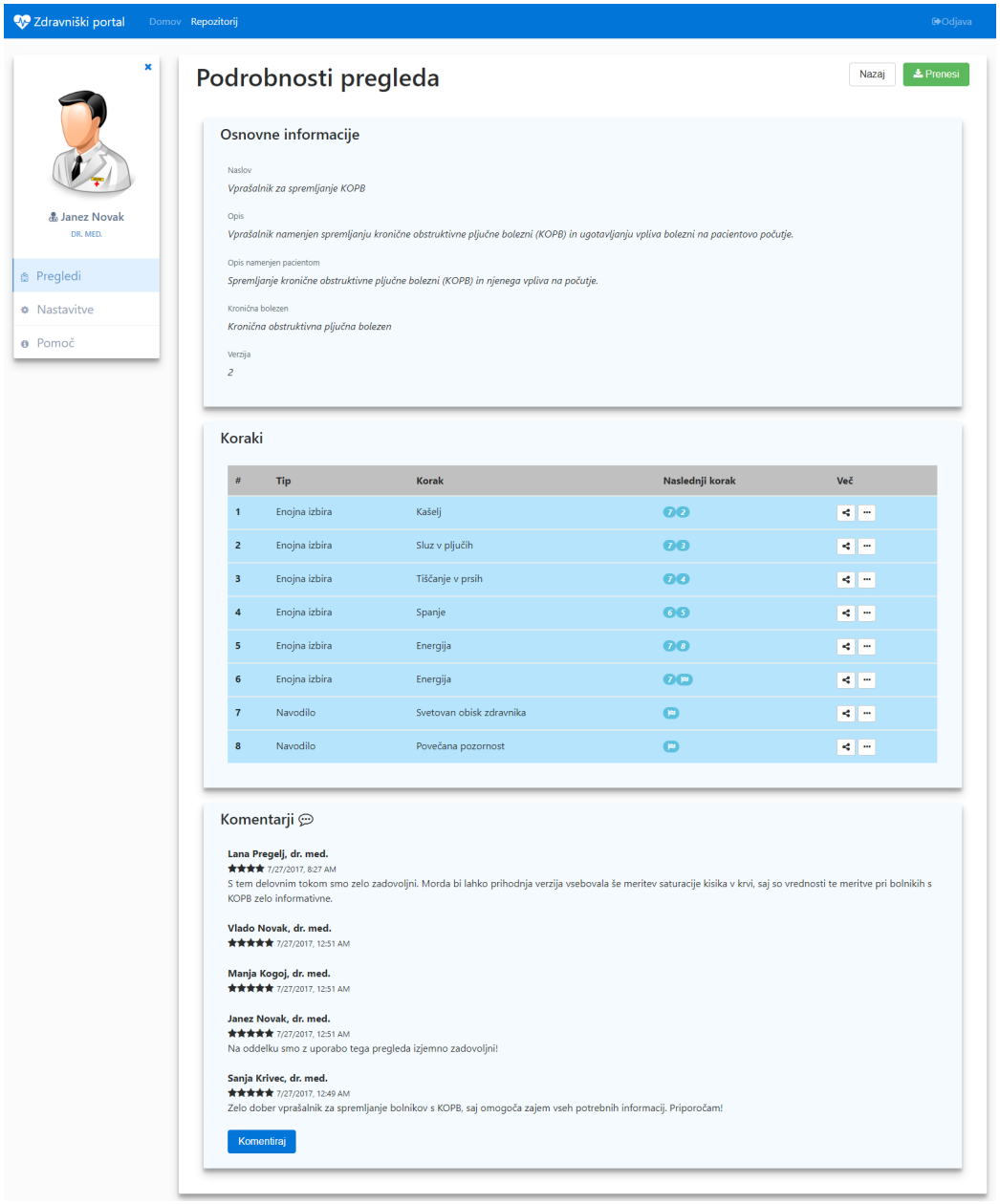
4.5.2 Globalni repozitorij pregledov



Slika 4.12: Seznam zdravniških pregledov v globalnem repozitoriju, namenjenih spremljanju kronične obstruktivne pljučne bolezni

Slika 4.12 predstavlja stran s seznamom zdravniških pregledov, ki so shranjeni v globalnem repozitoriju. Tako kot na strani pregledov iz lokalne podatkovne baze je tudi tu mogoče iskanje, filtriranje po kronični bolezni ter razvrščanje. Vsaka „kartica“ svetlo modre barve predstavlja eno procesno definicijo, kjer so v vrsticah prikazane posamezne verzije. Prikazan je tudi tip kronične bolezni, kateri je pregled namenjen. V vrsticah z verzijami sta skrajno desno podatka o številu prenosov procesne verzije in njena povprečna ocena. S klikom na eno izmed vrstic z izpisano verzijo se odpre okno 4.13.

Na oknu s podrobnostmi zdravniškega pregleda so prikazani trije razdelki. V prvem so predstavljene osnovne informacije procesne definicije, v drugem se nahaja tabela korakov, kjer se lahko s kliki na gumba v zadnjem stolpcu (*Več*) vidi še podrobne informacije koraka in njegove vejitve. Te informacije so prikazane v oknih, ki sta podobni tistim pri ustvarjanju pregleda, le da tu ni vnosnih polj, ampak so podatki le izpisani.

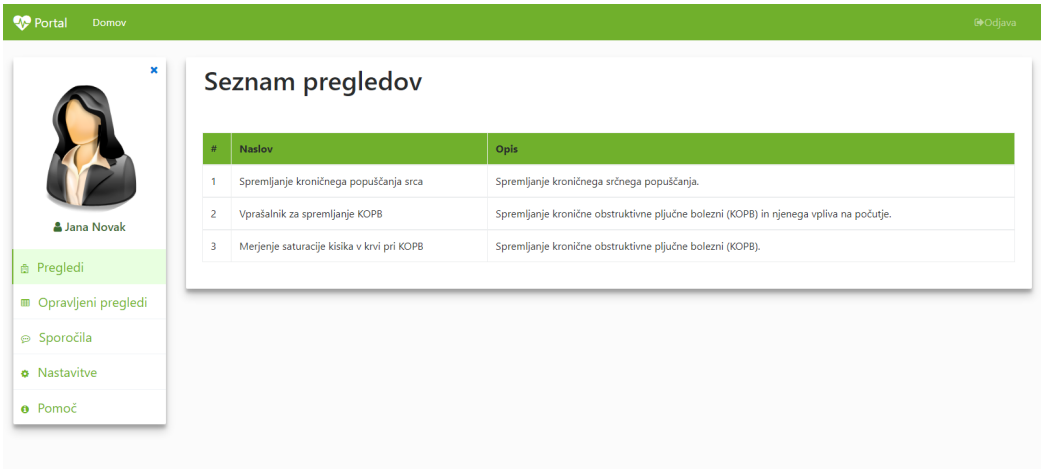


Slika 4.13: Pogled podrobnosti zdravniškega pregleda iz globalnega repozitorija

Zadnji razdelek pa vsebuje seznam ocen in komentarjev pregleda. Tega lahko namreč zdravniki ocenjujejo in obenem napišejo pozitivne ali negativne kri-

tike. V zgornjem desnem kotu okna se nahaja gumb *Prenesi*, ki omogoča prenos izbrane verzije zdravniškega pregleda iz repozitorija v lokalno podatkovno bazo.

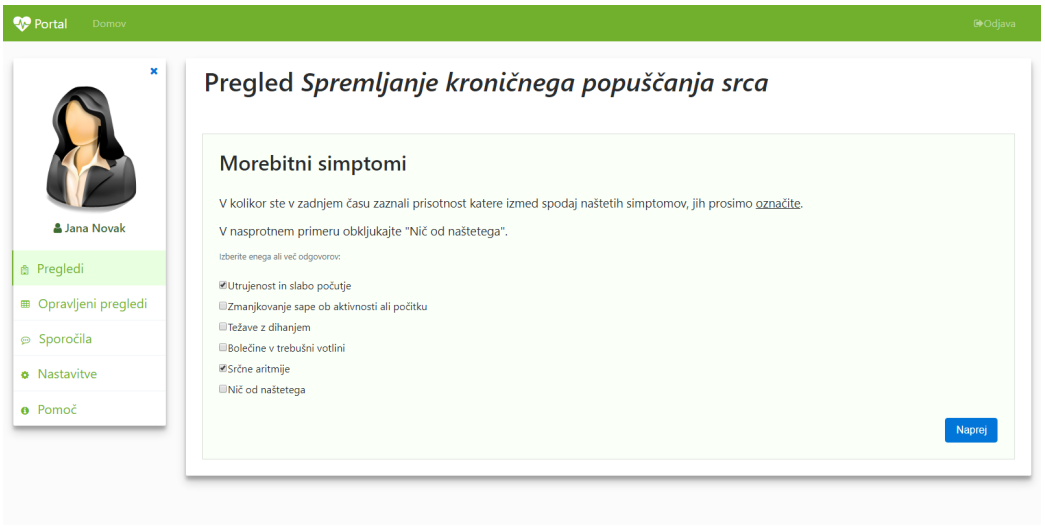
4.5.3 Aplikacija za opravljanje pregledov



Slika 4.14: Seznam zdravniških pregledov, ki jih mora opravljati nek pacient.

Glavna stran aplikacije za paciente (4.14) je sestavljena iz podobnih gradnikov kot začetna stran portala za ustvarjanje pregledov. Na levi strani se nahaja stranski menu, kjer lahko pacient izbira med različnimi podstranmi. Prikazana sta tudi ime in priimek bolnika ter morebitna profilna slika. Desno od menuja je okno, kjer se prikazuje glavna vsebina aplikacije. Na začetni strani je tabela zdravniških pregledov, ki jih mora pacient izpolnjevati. Ob izbiri enega izmed pregledov iz tabele je pacient preusmerjen na del aplikacije za izpolnjevanje tega pregleda. Prikaže se začetni korak, s klikom na *Nadaljuj* pa poteka premikanje naprej po pregledu. Primer koraka, ki vsebuje vprašanje tipa *Večkratna izbira*, je prikazan na sliki 4.15. Pacient lahko izbira med več ponujenimi vprašanji, vidno je tudi preprosto oblikovanje besedila. Ko se bolnik znajde na zadnjem koraku, se mu prikažeta besedilo o

zaključku pregleda in gumb *Zaključi pregled*, ki procesno instanco pošlje na strežnik, kjer se ta shrani.



Slika 4.15: Primer vprašanja z izbiro med več možnimi odgovori

4.6 Evalvacija rešitve

Če bi želeli dejansko ovrednotiti našo aplikacijo, bi jo bilo treba dati v preizkus večjemu številu zdravnikov, ki bi jo lahko stestirali in podali svoje mnenje. Le tako bi namreč lahko ugotovili, ali jim rešitev ustreza in kaj bi bilo potrebno še spremeniti ter nadgraditi. Ker pa ta način evalvacije presega namen diplomske naloge, smo se raje osredotočili na vrednotenje tehnične zmogljivosti razvitega orodja. Zato smo na spletu poiskali primer modela za spremljanje bolnikov z določeno kronično boleznijo na domu. Model smo vzeli kot osnovo, na podlagi katere smo v naši aplikaciji ustvarili zdravniški pregled za spremljanje bolnikov s to boleznijo. S tem smo želeli pokazati, da lahko zdravniki z uporabo razvite aplikacije ustvarjajo zdravniške preglede, ki omogočajo zajem meritev in odgovorov na vprašanja, ki so nujno potrebni za spremljanje neke bolezni. Sicer smo tudi preglede, ki so opisani ali prikazani v prejšnjih poglavjih, naredili na podlagi informacij, ki smo jih našli na

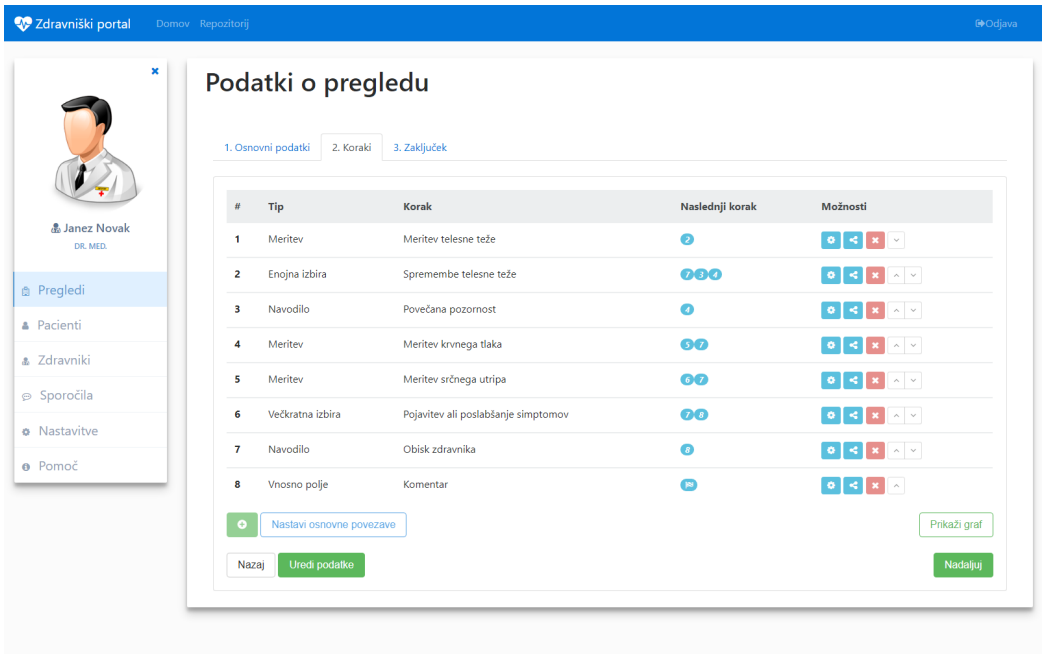
spletu, vendar so bile najdene informacije velikokrat nepopolne, zato je bilo potrebno dele pregledov prilagoditi ali dopolniti.

Poizkusili smo najti modele pregledov za različne tipe kroničnih bolezni, kot so npr. *diabetes*, *astma*, *kronična obstruktivna pljučna bolezen* in *kronično srčno popuščanje*. Ker smo najbolj natančne podatke o tem, katere vitalne znake je potrebno zajeti, našli za tip *kroničnega srčnega popuščanja*, obenem pa ta bolezen velja za eno izmed tistih, ki povzročijo največ smrti letno [43], smo se odločili ustvariti zdravniški pregled za njeno spremljanje. To bolezensko stanje nastane zaradi različnih okvar srca, kar prizadene njegovo sposobnost črpanja, to pa privede do težav pri črpanju krvi po telesu. Žal te kronične bolezni še ni mogoče povsem pozdraviti, z zdravili se lahko le zavira njeno napredovanje oz. se bolnikom izboljša kvaliteto življenja [11]. Da se to naredi čim bolj uspešno, je potreben čim boljši vpogled v zdravstveno stanje pacienta, ki se ga lahko pridobi z uporabo telemedicine.

Preučili smo več različnih virov [12, 43, 39] in iz njih razbrali, da se pri *kroničnem srčnem popuščanju* navadno spremlja naslednje vitalne znake: *telesno težo*, *krvni tlak* in *srčni utrip*. Pri enem viru je bila merjena še *koncentracija sladkorja v krvi*, saj je za veliko število bolnikov značilno, da imajo poleg kroničnega srčnega popuščanja tudi diabetes. Spremljanje telesne teže je pomembno, saj študije kažejo, da se tveganje za CHF pri moških poveča za pet odstotkov, pri ženskah pa za sedem, v primeru ko se BMI poveča le za en odstotek. Spremembe krvnega tlaka in srčnega utripa kažejo na težave s srčno-žilnim sistemom, zato je njuno spremljanje prav tako potrebno. Pojavitev ali poslabšanje naslednjih simptomov tudi lahko kaže na zaplete pri bolezni: *utrujenost*; *težave z dihanjem – ob aktivnosti ali v ležečem položaju*; *kašelj*; *napihovanje in zatekanje nog ali gležnjev*; *bolečine v prsih*; *razdražen želodec*; *povečanje pogostosti uriniranja, posebej ponoči*; *srčne aritmije* ...

Na podlagi omenjenih ugotovitev smo ustvarili zdravniški pregled za spremljanje bolnikov z diagnosticiranim kroničnim srčnim popuščanjem. Koraki iz katerih sestoji pregled, so prikazani na sliki 4.16. Sprva se opravi meritev telesne teže pacienta. Ker naša rešitev še ne implementira možnosti primer-

jave meritev z meritvami prejšnjih pregledov ali pa s podatki o pacientu (da bi torej lahko meritev teže primerjali s shranjenim podatkom o teži pacienta), je potreben še drugi korak, kjer pacient odgovori na vprašanje o spremembi teže. Bolnik mora tako sam vedeti, koliko je tehtal prej oz. mora to informacijo najti v zgodovini pregledov in vrednost primerjati s pravkar izmerjeno. Če se je masa povečala za več kot kilogram in pol v enem dnevu ali za več kot dva kilograma v enem tednu, se zgodi preusmeritev na sedmi korak, kjer se pacientu svetuje čimprejšen obisk zdravnika. Tako hitro povečevanje teže namreč nakazuje na zaostrovanje težav, saj prihaja do zastajanja tekočine v telesu. V kolikor se je telesna teža znižala, se prikaže obvestilo (tretji korak), naj bo pacient v prihodnje še bolj pozoren na svojo težo in naj v primeru velikih odstopanj obišče zdravnika. Hitra izguba telesne teže lahko namreč prav tako nakazuje na težave.



Slika 4.16: Primer zdravniškega pregleda za spremljanje kroničnega srčnega popuščanja

Iz tega koraka in pa v primeru, da ni prišlo do večjih sprememb, se napreduje

na meritev krvnega tlaka. Če je sistolični tlak nižji od 150 mmHg in višji od 90 mmHg ter je obenem diastolični tlak nižji od 90 mmHg in višji od 50 mmHg, sledi meritev srčnega utripa. V nasprotnem primeru so vrednosti krvnega tlaka izven predvidenih meja, zato se svetuje obisk zdravnika. Obisk je svetovan tudi, če je srčni utrip višji od 90 udarcev na minuto ali nižji od 40 udarcev od minuto. Sicer pa se pacientu prikaže vprašanje, kjer ima na voljo izbiro med več odgovori. Prikazani so prej navedeni simptomi, označiti pa je potrebno tiste, ki so se na novo pojavili oz. pri katerih je prišlo do poslabšanja. Če so se pojavile spremembe pri enem ali več simptomih, lahko to nakazuje zaplete, zato se predlaga posvet z zdravnikom. Zadnji korak, ki se v vsakem primeru prikaže pacientu, je vnosno polje, kamor mora vpisati kakršenkoli komentar, s katerim lahko opiše svoje počutje, dnevne aktivnosti ali pa pojasni, zakaj je morda prišlo do takih vrednosti meritev, kot so bile izmerjene.

S predstavljenim primerom nam je uspelo demonstrirati, da bi se aplikacijo za ustvarjanje zdravniških pregledov, ki bi sledila principom uporabljenim v naši rešitvi, lahko uporabljalo v praksi, saj je z njo mogoče ustvarjati preglede, ki zajemajo vse vitalne znake, ki so potrebni za spremljanje bolnika na domu. Tako ima lahko zdravnik podroben nadzor nad pacientovim zdravstvenim stanjem, ne da bi bilo temu treba pogosto obiskovati zdravstveno ustanovo.

4.6.1 Izboljšave

Preden bi lahko razvito rešitev v celoti postavili kot zgled aplikacijam, katerih namen je modeliranje zdravniških pregledov, je potrebnih še kar nekaj nadgradenj in izboljšav.

Implementirati bi bilo treba del za upravljanje z uporabniki (tako z zdravniki neke zdravstvene ustanove kot tudi s pacienti) ter sprogramirati ustrezen način registracije in prijave. Tako bi procesne definicije vsebovale še podatek o tem, kateri zdravnik jih je ustvaril. Le-ta bi pacientom tudi določal, katere zdravniške preglede morajo opravljati. Dodana bi morala biti tudi funkcio-

nalnost, s katero bi zdravniki definirali, kdaj se mora nek pregled opraviti; naprava, na kateri bi tekla aplikacija za paciente, pa bi prikazala opozorilo, ko bi nastopil čas za izpolnjevanje. Zelo pomembna je tudi povezava procesnih definicij in samih korakov pregleda s pacienti. To pomeni, da bi se lahko tako vsakemu bolniku posebej, kot tudi npr. skupinam bolnikov z istim bolezenskim stanjem, določilo meje meritev na način, kot ga implementira *Opentelehealth*. Tako bi se pri vejitvah namesto dejanskih vrednosti (kot smo to npr. naredili pri meritvah krvnega tlaka in srčnega utripa pri ustvarjenem zdravniškem pregledu za spremljanje kroničnega srčnega popuščanja), določilo, naj bo npr. srčni utrip višji od x udarcev na minuto in nižji od y udarcev na minuto, kjer bi bili meji x in y določeni za vsakega bolnika posebej. S tem je mogoče preglede na enostaven način bolj prilagoditi posamezniku, saj se „dovoljene“ vrednosti meritev ponavadi razlikujejo od pacienta do pacienta. Za nekoga s hudo okvaro srca je zagotovo zaželjeno, da vzdržuje nižji srčni utrip kot nekdo, ki ga pesti lažja oblika kake druge bolezni. S povezavo meritev in bolniku določenih podatkov bi imeli tudi možnost skrajšati in poenostaviti pregled za spremljanje kroničnega srčnega popuščanja; drugo vprašanje namreč ne bi bilo več potrebno, saj bi se lahko že pri prvem koraku naredilo primerjavo meritve telesne teže s težo, shranjeno v pacientovem profilu.

Prav tako bi zdravnikom lahko še bolj olajšali delo, če bi ustvarjanju zdravniških pregledov dodali možnost predogleda, s čimer bi bilo mogoče tekom modeliranja preveriti, kako bo proces izgledal na pacientovi napravi. Na strežniški in odjemalski strani aplikacije bi morali podpreti uporabo v diplomskemu nalogi že predstavljenega tipa koraka, ki predstavlja množico meritev ter vprašanj in kateremu je bila prilagojena struktura podatkovne baze. Zdravniškemu portalu bi se implementiralo še dodatne tipe vprašanj, ki bi olajšali ustvarjanje pregledov in hkrati omogočali, da se iz zajetih podatkov pridobi še več informacij. Prvi takšen tip je npr. vprašanje, ki ima na voljo odgovora „da“ in „ne“, in ki se ga trenutno lahko ustvari z uporabo tipa *Enojna izbira*, a je dodajanje zamudno. Drugi tip pa predstavlja

vprašanje, kjer se lahko pripravljene primere trditev ocenjuje oz. se jim določa številsko vrednost. Na koncu koraka pa se točke sešteje in se na podlagi seštevka definira vejitve. Tak tip bi lahko bil uporabljen za šesti korak pri pregledu za bolnike s kroničnim srčnim popuščanjem, kjer so navedeni simptomi, ki jih mora pacient ob pojavitvi oz. poslabšanju izbrati. Pri enem izmed preučenih virov [43] so namreč pacienti za vsak naveden simptom morali izbrati številsko vrednost od nič do tri, kjer je nič pomenila, da ta simptom pri njih ni prisoten, tri pa, da imajo z njim velike težave. Glede na to, kolikšno število točk so nato bolniki zbrali pri tem vprašanju, se je lahko določilo izboljšanje ali poslabšanje bolezenskega stanja. Predvsem za paciente bi bilo dobro implementirati še tipe, ki bi npr. omogočali izbiro številske vrednosti z drsnikom, izbiro dneva s koledarja idr., saj se na ta način naredi reševanje bolj intuitivno, hkrati pa se zmanjša možnost za napake. Če se nam pri oblikovanju formata pregledov za izmenjavo ni uspelo držati standarda FHIR, pa bi lahko meritvam določili kode LOINC in s tem naredili izmenjavo pregledov med različnimi sistemi še bolj učinkovito.

Predstavili smo le nekaj nadgradenj in izboljšav, ki bi precej pripomogle k bolj učinkoviti in enostavnejši rabi naše rešitve. Seveda bi se lahko implementiralo še druge funkcionalnosti npr. v povezavi z lepšim in bolj informativnim prikazovanjem zajetih podatkov, njihovemu združevanju in učenju nad njimi. Možnosti je veliko, a ponavadi njihova implementacija ni ravno trivialna.

Poglavje 5

Sklep

V diplomski nalogi smo zasnovali in razvili spletno aplikacijo, ki zdravnikom omogoča ustvarjanje zdravniških pregledov, namenjenih oddaljenemu spremljanju pacientov. V začetnih poglavjih smo analizirali obstoječe spletne rešitve z istim ali podobnim namenom, jih med seboj primerjali in definirali ključne funkcionalnosti, ki jih mora naša aplikacija implementirati. Osredotočili smo se predvsem na način ustvarjanja pregledov, njihovo shranjevanje ter izmenjavo. Na podlagi primerjave in lastnih ugotovitev smo nato načrtovali obliko zdravniških pregledov. V naslednjih poglavjih smo predstavili arhitekturo aplikacije in opisali posamezne komponente, ki jo sestavljajo. Začeli smo s podatkovnim modelom in opisom tabel podatkovne baze, opisali predstavitev pregledov v formatu JSON, potem pa se posvetili zalednim sistemom in odjemalskemu delu. Ker arhitektura aplikacije temelji na storitvah REST, smo predstavili vire, ki jih izpostavljajo zaledni sistemi, pri čemer smo poleg splošnih informacij podali tudi nekaj konkretnih primerov programske kode. Veliko pozornosti smo namenili tudi predstavitvi shranjevanja zdravniških pregledov, pretvarjanju vprašalnikov FHIR in implementaciji globalnega repozitorija. Nato smo se osredotočili na odjemalski del, kjer smo dodali tudi primere zaslonskih mask ter s tem predstavili izgled uporabniškega vmesnika in pokazali, kako se aplikacijo uporablja. Na koncu smo na spletu poiskali in preučili še model enega izmed zdravniških

pregledov, ki se v realnosti uporabljajo za oddaljeno spremljanje pacientov z določeno kronično boleznijo. Na podlagi modela smo z našim orodjem ustvarili zdravniški pregled ter s tem v praksi demonstrirali možnost uporabe principov, ki jim sledi naša aplikacija.

Z implementacijo orodja za ustvarjanje zdravniških pregledov, namenjenih oddaljenemu spremljanju pacienta, in njegovo predstavitvijo v diplomski nalogi, smo prikazali smernice, ki jih je priporočljivo upoštevati pri razvoju aplikacij s takim namenom. Ustvarjena rešitev, ki ji je sicer potrebno še implementirati dodatne nadgradnje in izboljšave, omenjene v poglavju 4.6.1, tako služi kot zgled aplikacijam za modeliranje zdravniških pregledov.

Glavne cilje, ki smo si jih zastavili pred začetkom izdelave diplomske naloge, smo v veliki meri dosegli, saj nam je uspelo načrtovati in izdelati delujočo spletno aplikacijo za modeliranje zdravniških pregledov. Dodali smo tudi funkcionalnost, ki omogoča shranjevanje pregledov v globalni repozitorij in prenašanje le-teh iz njega. Žal nam zaradi izzivov, navedenih v poglavju 4.4.3, ni uspelo podpreti izmenjave zdravniških pregledov v formatu, ki upošteva standard FHIR.

Nadaljni koraki, s katerimi bi še povečali težo diplomskega dela, bi zajemali implementacijo nadgradenj in testiranje uporabe aplikacije pri končnih uporabnikih (zdravnikih). Prav njihovi odzivi in kritike bi nam ponudili ključne informacije, potrebne za dodatne izboljšave in morebitne spremembe spletne aplikacije. Pravzaprav bi morali, če bi bilo to časovno izvedljivo, končne uporabnike vključiti v razvoj že prej. Zajem njihovih zahtev in želja bi omogočil, da bi bila implementacija že v tej fazi razvoja še bolj prilagojena njihovim potrebam.

Tekom izdelave diplomske naloge smo pridobili veliko izkušenj iz načrtovanja in izdelave spletnih aplikacij ter oblikovanja uporabniških vmesnikov. Hkrati smo se priučili uporabe novih tehnologij, kot sta Angular 4 in Node.js, s katerima prej nismo imeli izkušenj, izbrali pa smo ju predvsem z namenom, da pridobimo nova znanja o modernih tehnologijah, ki se uporabljajo za razvoj spletnih aplikacij.

Literatura

[1] Alayacare documentation. Dosegljivo: http://www.alayacare.com/. [Dostopano 25.06.2017].

[2] Alayacare's remote patient monitoring solution. Dosegljivo: https://www.youtube.com/watch?v=eclQ8hii0eQ. [Dostopano 25.06.2017].

[3] Angular 4. Dosegljivo: https://angular.io/. [Dostopano 25.07.2017].

[4] Bitbucket. Dosegljivo: https://bitbucket.org/. [Dostopano 26.07.2017].

[5] Bootstrap v4. Dosegljivo: https://v4-alpha.getbootstrap.com/. [Dostopano 25.07.2017].

[6] Care worker mobile app. Dosegljivo: https://www.youtube.com/watch?v=m0AwBjLN-t0. [Dostopano 25.06.2017].

[7] Extensibility. Dosegljivo: https://www.hl7.org/fhir/extensibility.html. [Dostopano 17.07.2017].

[8] Fhir overview - developers. Dosegljivo: https://www.hl7.org/fhir/overview-dev.html. [Dostopano 16.07.2017].

[9] Git. Dosegljivo: https://git-scm.com/. [Dostopano 25.07.2017].

[10] Global health observatory (gho) data - life expentacy. Dosegljivo: http://www.who.int/gho/mortality_burden_disease/life_tables/situation_trends_text/en/. [Dostopano 04.07.2017].

[11] Heart disease and congestive heart failure. Dosegljivo: http://www.webmd.com/heart-disease/guide-heart-failure#4. [Dostopano 28.07.2017].

[12] Heart failure - home monitoring. Dosegljivo: https://medlineplus.gov/ency/patientinstructions/000113.htm. [Dostopano 28.07.2017].

[13] Honeywell genesis touch. Dosegljivo: https://www.youtube.com/watch?v=IOgDVA9_lL8. [Dostopano 28.06.2017].

[14] Honeywell lifecare products. Dosegljivo: https://www.honeywelllifecare.com/lifestream-products/. [Dostopano 28.06.2017].

[15] Loinc. Dosegljivo: https://loinc.org/. [Dostopano 17.07.2017].

[16] Medable documentation. Dosegljivo: https://docs.medable.com/. [Dostopano 1.07.2017].

[17] Node.js. Dosegljivo: https://nodejs.org/en/. [Dostopano 25.07.2017].

[18] Opentelehealth. Dosegljivo: http://opentelehealth.com/. [Dostopano 1.07.2017].

[19] Paperless homecare. Dosegljivo: https://www.youtube.com/watch?v=ZP4QufXZU_M. [Dostopano 25.06.2017].

[20] Phillips ecarecompanion. Dosegljivo: http://www.usa.philips.com/healthcare/product/HCNOCTN483. [Dostopano 28.06.2017].

[21] Phillips ecarecoordinator. Dosegljivo: http://www.usa.philips.com/healthcare/product/HCNOCTN482/ecarecoordinator-clinical-dashboard-for-ambulatory-health. [Dostopano 28.06.2017].

[22] Phillips healthsuite. Dosegljivo: http://www.usa.philips.com/healthcare/innovation/about-health-suite. [Dostopano 28.06.2017].

[23] Postgresql. Dosegljivo: https://www.postgresql.org/. [Dostopano 25.07.2017].

[24] Report: Iot in healthcare will top $163b by 2020. Dosegljivo: http://www.mobihealthnews.com/content/report-iot-healthcare-will-top-163b-2020. [Dostopano 03.07.2017].

[25] Researchkit. Dosegljivo: https://www.apple.com/researchkit/. [Dostopano 1.07.2017].

[26] Resource questionnaire. Dosegljivo: https://www.hl7.org/fhir/questionnaire.html. [Dostopano 16.07.2017].

[27] Resource questionnaire - examples. Dosegljivo: https://www.hl7.org/fhir/questionnaire-examples.html. [Dostopano 16.07.2017].

[28] Resource questionnaireresponse. Dosegljivo: https://www.hl7.org/fhir/questionnaireresponse.html. [Dostopano 16.07.2017].

[29] Tactio health. Dosegljivo: http://www.tactiohealth.com/tactiorpm. [Dostopano 27.06.2017].

[30] Tactio health devices. Dosegljivo: http://www.tactiohealth.com/devices/. [Dostopano 27.06.2017].

[31] Tinymce. Dosegljivo: https://www.tinymce.com/. [Dostopano 21.07.2017].

[32] Vivify. Dosegljivo: http://www.vivifyhealth.com/. [Dostopano 27.06.2017].

[33] Vivify go. Dosegljivo: http://www.vivifyhealth.com/products/go/. [Dostopano 27.06.2017].

[34] Vivify home. Dosegljivo: http://www.vivifyhealth.com/products/pathways-home/. [Dostopano 27.06.2017].

[35] Zephyrlife™ home remote patient monitoring brochure. Dosegljivo: http://www.medtronic.com/content/dam/covidien/library/us/en/product/health-informatics-and-monitoring/zephyr-home-brochure.pdf. [Dostopano 1.07.2017].

[36] Zephyrlife™ product details brochure. Dosegljivo: http://www.medtronic.com/content/dam/covidien/library/us/en/product/health-informatics-and-monitoring/zephyr-home-product-details.pdf. [Dostopano 1.07.2017].

[37] Martin Burwitz, Hannes Schlieter, and Werner Esswein. Modeling clinical pathways-design and application of a domain-specific modeling language. In *Wirtschaftsinformatik*, page 83, 2013.

[38] Reinhard Busse. *Tackling chronic disease in Europe: strategies, interventions and challenges*. Number 20. WHO Regional Office Europe, 2010.

[39] Cathy A Eastwood, Lucille Travis, Tanya T Morgenstern, and Erin K Donaho. Weight and symptom diary for self-monitoring in heart failure clinic patients. *Journal of Cardiovascular Nursing*, 22(5):382–389, 2007.

[40] Ed Eurostat. The life of women and men in europe: A statistical portrait. *Luxembourg: European Commission*, pages 15–16, 2008.

[41] Breda Hajnrih, Saša Kadivec, Zdenka Kramar, Dorjan Marušič, Tanja Mate, Mircha Poldrugovac, Valentina Prevolnik Rupel, Biserka Simčič, and Anne-Marie Yazbeck. *Priročnik za oblikovanje kliničnih poti*. Ministrstvo za zdravje, 2009.

[42] Joseph Kvedar, Molly Joel Coye, and Wendy Everett. Connected health: a review of technologies and strategies to improve patient care with telemedicine and telehealth. *Health Affairs*, 33(2):194–199, 2014.

[43] Myung-kyung Suh, Chien-An Chen, Jonathan Woodbridge, Michael Kai Tu, Jung In Kim, Ani Nahapetian, Lorraine S Evangelista, and Majid Sarrafzadeh. A remote patient monitoring system for congestive heart failure. *Journal of medical systems*, 35(5):1165–1179, 2011.

[44] Richard Wootton. Twenty years of telemedicine in chronic disease management–an evidence synthesis. *Journal of telemedicine and telecare*, 18(4):211–220, 2012.